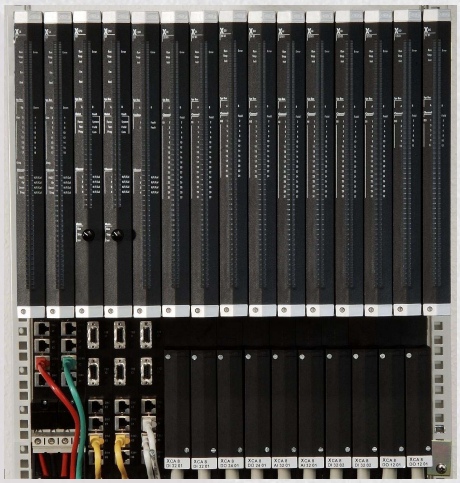


HIMax®

Kommunikationshandbuch

SAFETY
NONSTOP

HIMA

KOMMUNIKATION

8

Alle in diesem Handbuch genannten HIMA Produkte sind mit dem Warenzeichen geschützt. Dies gilt ebenfalls, soweit nicht anders vermerkt, für weitere genannte Hersteller und deren Produkte.

Alle technischen Angaben und Hinweise in diesem Handbuch wurden mit größter Sorgfalt erarbeitet und unter Einschaltung wirksamer Kontrollmaßnahmen zusammengestellt. Bei Fragen wenden Sie sich bitte direkt an HIMA. Für Anregungen, z. B. welche Informationen noch in das Handbuch aufgenommen werden sollen, ist HIMA dankbar.

Technische Änderungen vorbehalten. Ferner behält sich HIMA vor, Aktualisierungen des schriftlichen Materials ohne vorherige Ankündigungen vorzunehmen.

Weitere Informationen sind in der Dokumentation auf der CD-ROM und auf unserer Webseite unter http://www.hima.de und http://www.hima.com zu finden.

© Copyright 2009, HIMA Paul Hildebrandt GmbH + Co KG

Alle Rechte vorbehalten.

Kontakt

Die HIMA Adresse ist:

HIMA Paul Hildebrandt GmbH + Co KG

Postfach 1261

D-68777 Brühl

Tel: +49 6202 709-0

Fax: +49 6202 709-107

E-Mail: info@hima.com

Inhaltsverzeichnis

1 Einleitung 11

1.1 Aufbau und Gebrauch des Handbuchs 11

1.2 Zielgruppe 12

1.3 Darstellungskonventionen 12

1.3.1 Sicherheitshinweise 12

1.3.2 Gebrauchshinweise 13

2 Sicherheit 14

2.1.1 Einsatzbedingungen 14

2.2 Restgefahren 17

2.3 Sicherheitsvorkehrungen 17

2.4 Notfallinformation 17

3 Produktbeschreibung 18

3.1 Sicherheitsgerichtetes Protokoll (safeethernet) 18

3.2 Standardprotokolle 19

3.3 Redundanz 20

3.4 Aufbau der HIMax COM-Modul Teilenummer 21

3.5 Registrierung und Aktivierung der Protokolle 22

3.6 Ethernet-Schnittstellen 23

3.6.1 Eigenschaften der Ethernet-Schnittstellen 23

3.6.2 Konfiguration der Ethernet-Schnittstellen 24

3.6.3 Verwendete Netzwerkports für Ethernet-Kommunikation 29

3.7 Feldbusschnittstellen 29

3.7.1 Pin-Belegung der D-SUB-Buchsen FB1 und FB2 30

4 safeethernet 31

4.1 Was ist safeethernet 32

4.2 Konfiguration einer redundanten safeethernet Verbindung 34

4.2.1 Prozessvariablen verbinden 35

4.2.2 Prüfung der safeethernet Kommunikation 36

4.3 safeethernet-Editor 37

4.4 Detailansicht des safeethernet-Editors 39

4.4.1 Register: Systemvariablen 39

4.5 Mögliche safeethernet Verbindungen 42

4.5.1 Mono safeethernet Verbindung (Kanal 1) 42

4.5.2 Redundante safeethernet Verbindung (Kanal 1 und Kanal 2) 42

4.5.3 Erlaubte Kombinationen 43

4.6 safeethernet Parameter 44

4.6.1 Maximale Zykluszeit (minimale Watchdog-Zeit) der HIMax Steuerung 44

4.6.2 Receive Timeout 45

4.6.3 ResponseTime 45

4.6.4 Sync/Async 46

4.6.5 ResendTMO 46

4.6.6 Acknowledge Timeout 46

4.6.7 Production Rate 47

4.6.8 Speicher 47
4.7 Maximale Reaktionszeit für safeethernet 48
4.7.1 Berechnung der maximalen Reaktionszeit zweier HIMax PES 48
4.7.2 Berechnung der max. Reaktionszeit in Verbindung mit einem HIMatrix-PES 49
4.7.3 Berechnung der max. Reaktionszeit mit zwei HIMatrix-Steuerungen oder RIOs 49
4.7.4 Berechnung der max. Reaktionszeit mit zwei HIMax und einem HIMatrix-PES 50
4.7.5 Safeethernet Profile 51
4.7.6 Profil I (Fast & Cleanroom) 52
4.7.7 Profil II (Fast & Noisy) 52
4.7.8 Profil III (Medium & Cleanroom) 53
4.7.9 Profil IV (Medium & Noisy) 53
4.7.10 Profil V (Slow & Cleanroom) 54
4.7.11 Profil VI (Slow & Noisy) 54
4.8 Projektübergreifende Kommunikation 55
4.8.1 Varianten zur projektübergreifenden Kommunikation 56
4.9 Projektübergreifende Kommunikation zwischen SILworX und ELOP II Factory 57
4.9.1 Konfiguration der HIMax im SILworX Projekt 57
4.9.2 Konfiguration der HIMatrix in ELOP II Factory 61
4.10 Control Panel (safeethernet) 63
4.10.1 Anzeigefeld (safe**ethernet**-Verbindung) 64
5 PROFINET-IO 65
5.1 PROFINET-IO Funktionsbausteine 65
5.2 HIMA PROFINET-IO Controller 66
5.3 Systemanforderung 66
5.4 PROFINET-IO Beispiel 67
5.4.1 Anlegen des HIMA PROFINET-IO Controller in SILworX 67
5.5 Menüfunktion im PROFINET-IO Controller 69
5.5.1 Eigenschaften 69
5.6 Menüfunktionen PROFINET-IO Device (im Controller) 70
5.6.1 Eigenschaften 70
5.6.2 DAP Modul (Device Access Point Modul) 71
5.6.3 Input/Output PROFINET-IO Module 71
5.6.4 Submodul Input 72
5.6.5 Submodul Output 73
5.6.6 Submodul Inputs und Outputs 75
5.6.7 Application Relation 77
5.6.8 Alarm CR 78
5.6.9 Input CR 79
5.6.10 Output CR 80
5.7 HIMA PROFINET-IO Device 81
5.8 Systemanforderung 81
5.9 PROFINET-IO Beispiel 82
5.9.1 Konfiguration des PROFINET-IO Device in SILworX 82
5.9.2 Anlegen des HIMA PROFINET-IO Controller in SILworX 85
5.9.3 Menüfunktion Eigenschaften 89
5.9.4 PROFINET-IO Module 90

6 PROFIBUS DP 92

6.1 HIMA PROFIBUS DP Master 93
6.1.1 Anlegen eines HIMA PROFIBUS DP Master 93

6.2 PROFIBUS DP Beispiel 94
6.2.1 PROFIBUS DP Slave Konfigurieren 94
6.2.2 PROFIBUS DP Master Konfigurieren 96

6.3 Menüfunktionen des PROFIBUS DP Master 103
6.3.1 Edit 103
6.3.2 Menüfunktion Eigenschaften 104

6.4 Die PROFIBUS-DP Buszugriffsverfahren 108
6.4.1 Master/Slave-Protokoll 108
6.4.2 Token-Protokoll 108
6.4.3 Token-Umlaufzeit (Ttr) 108
6.4.4 Token-Umlaufzeit Ttr berechnen 109

6.5 Isochroner PROFIBUS DP Zyklus (ab DP-V2) 111
6.5.1 Isochron Mode (ab DP-V2) 112
6.5.2 Isochron Mode Sync (ab DP-V2) 112
6.5.3 Isochrone Mode Freeze (ab DP-V2) 112

6.6 Menüfunktionen PROFIBUS DP Slave (im Master) 113
6.6.1 Anlegen eines PROFIBUS DP Slave (im Master) 113
6.6.2 Edit 113
6.6.3 Eigenschaften 114

6.7 GSD-Datei einlesen 119

6.8 User-Parameter bearbeiten 120

6.9 PROFIBUS Funktionsbausteine 122
6.9.1 Funktionsbaustein MSTAT 123
6.9.2 Funktionsbaustein RALRM 126
6.9.3 Funktionsbaustein RDIAG 130
6.9.4 Funktionsbaustein RDREC 134
6.9.5 Funktionsbaustein SLACT 137
6.9.6 Funktionsbaustein WRREC 140

6.10 PROFIBUS Hilfsfunktionsbausteine 143
6.10.1 Hilfsfunktionsbaustein ACTIVE 143
6.10.2 Hilfsfunktionsbaustein ALARM 144
6.10.3 Hilfsfunktionsbaustein DEID 145
6.10.4 Hilfsfunktionsbaustein ID 146
6.10.5 Hilfsfunktionsbaustein NSLOT 147
6.10.6 Hilfsfunktionsbaustein SLOT 147
6.10.7 Hilfsfunktionsbaustein STDDIAG 148

6.11 Fehlercodes der Funktionsbausteine 150

6.12 Control-Panel (PROFIBUS DP Master) 151
6.12.1 Kontextmenü (PROFIBUS Master) 151
6.12.2 Anzeigefeld (PROFIBUS Master) 152
6.12.3 Zustand des PROFIBUS DP Master 153
6.12.4 Verhalten des PROFIBUS DP Master 153
6.12.5 Funktion der FBx LED beim PROFIBUS DP Master 154

6.13 HIMA PROFIBUS DP Slave 155
6.13.1 Anlegen eines HIMA PROFIBUS DP Slave 155

6.14 Menüfunktionen PROFIBUS DP Slave 156
6.14.1 Edit 156
6.14.2 Eigenschaften 158
6.15 Control-Panel (PROFIBUS DP Slave) 160
6.15.1 Kontextmenü (PROFIBUS DP Slave) 160
6.15.2 Anzeigefeld (PROFIBUS DP Slave) 160
6.16 Funktion der FBx LED beim PROFIBUS DP Slave 161
7 Modbus 162
7.1 HIMA Modbus Master 163
7.2 Modbus Beispiel 164
7.2.1 Konfiguration des Modbus TCP Slave 164
7.2.2 Konfiguration des Modbus TCP Master 166
7.3 Beispiel zur Alternativen Register/Bit-Adressierung 168
7.4 Menüfunktionen des HIMA Modbus Master 169
7.4.1 Edit 169
7.4.2 Eigenschaften 170
7.5 Modbus Funktionscodes (Anforderungstelegramme) 172
7.5.1 Modbus Standard Funktionscodes 172
7.5.2 HIMA spezifische Funktionscodes 173
7.5.3 Anforderungstelegramme zum Lesen 175
7.5.4 Anforderungstelegramm zum Schreiben 177
7.5.5 Ethernet Slaves (TCP/UDP-Slaves) 179
7.5.6 Systemvariablen der TCP/UDP-Slaves 180
7.5.7 Eigenschaften TCP/UDP-Slaves 180
7.5.8 Modbus Gateway (TCP/UDP Gateway) 182
7.5.9 Eigenschaften Gateway 184
7.5.10 Systemvariablen Gateway-Slave 184
7.5.11 Eigenschaften Gateway-Slave 184
7.5.12 Serieller Modbus 185
7.5.13 Eigenschaften Serieller Modbus 186
7.5.14 Systemvariablen Modbus Slave 186
7.5.15 Eigenschaften Modbus Slave 187
7.6 Control-Panel (Modbus Master) 187
7.6.1 Kontextmenü (Modbus Master) 188
7.6.2 Anzeigefeld (Modbus Master) 188
7.7 Control-Panel (Modbus Master->Slave) 188
7.8 Funktion der FBx LED beim Modbus Master 189
7.9 HIMA Modbus Slave 190
7.9.1 Konfiguration des Modbus TCP Slave 191
7.9.2 Konfiguration des redundanten Modbus TCP Slave 191
7.9.3 Regeln für den redundanten Modbus TCP Slave 192
7.10 Menüfunktionen des HIMA Modbus Slave Set 193
7.10.1 Eigenschaften Modbus Slave Set 193
7.10.2 Registervariable 194
7.10.3 Bitvariablen 194
7.10.4 Sende/Empfangsvariablen zuweisen 195
7.10.5 Systemvariablen Modbus Slave Set 195
7.10.6 Modbus Slave und Modbus Slave Redundant 195
7.10.7 Modbus-Funktionscodes 198

7.10.8 HIMA spezifischen Funktionscodes ... 200
7.11 Modbus Adressierung durch Bit und Register ... 202
7.11.1 Register Bereich ... 202
7.11.2 Bit Bereich ... 203
7.12 Offsets für alternative Modbus Adressierung ... 204
7.12.1 Zugriff auf die Registervariablen im Bit Bereich des Modbus Slave ... 205
7.12.2 Zugriff auf die Bitvariablen im Registerbereich des Modbus Slave ... 206
7.13 Control-Panel (Modbus Slave) ... 207
7.13.1 Kontextmenü (Modbus Slave) ... 207
7.13.2 Anzeigefeld (Modbus Slave) ... 208
7.13.3 Anzeigefeld (Masterdaten) ... 208
7.14 Funktion der FBx LED beim Modbus Slave ... 209
7.14.1 Fehlercodes der Modbus TCP/IP-Verbindung ... 209
8 Send & Receive TCP ... 210
8.1 Systemanforderung ... 210
8.1.1 Anlegen eines S&R-TCP-Protokolls ... 210
8.2 Beispiel: S&R TCP Konfiguration ... 211
8.2.1 S&R TCP Konfiguration der Siemens Steuerung SIMATIC 300 ... 213
8.2.2 S&R TCP Konfiguration der HIMax Steuerung ... 217
8.3 Menüfunktionen S&R-TCP-Protokoll ... 219
8.3.1 Edit ... 219
8.3.2 Eigenschaften ... 219
8.4 Menüfunktionen TCP Verbindung ... 221
8.4.1 Edit ... 221
8.4.2 Systemvariablen ... 221
8.4.3 Eigenschaften ... 222
8.5 Datenaustausch ... 224
8.5.1 TCP-Verbindungen ... 224
8.5.2 Zyklischer Datenaustausch ... 225
8.5.3 Azyklischer Datenaustausch mit Funktionsbausteine ... 225
8.5.4 Gleichzeitiger zyklischer und azyklischer Datenaustausch ... 225
8.5.5 Flusskontrolle ... 226
8.6 Fremdsysteme mit Pad Bytes ... 226
8.7 S&R-TCP-Funktionsbausteine ... 227
8.7.1 TCP_Reset ... 228
8.7.2 TCP_Send ... 231
8.7.3 TCP_Receive ... 234
8.7.4 TCP_ReceiveLine ... 238
8.7.5 TCP_ReceiveVar ... 242
8.8 Control-Panel (Send/Receive over TCP) ... 247
8.8.1 Kontextmenü (Send/Receive Protokoll) ... 247
8.8.2 Anzeigefeld (Send/Receive Protokoll) ... 247
8.8.3 Anzeigefeld (Send/Receive Server) ... 247
8.8.4 Fehlercode der TCP-Verbindung ... 248
8.8.5 Zusätzliche Fehlercodetabelle der Funktionsbausteine ... 249
8.8.6 Verbindungszustand ... 249
8.8.7 Partner Verbindungszustand ... 249

9 SNTP-Protocol 250

9.1 SNTP-Client 250

9.2 SNTP-Client (Server Info) 252

9.3 SNTP-Server 253

10 X-OPC-Server 254

10.1 Benötigte Ausstattung und Systemanforderung 254

10.2 Eigenschaften des X-OPC-Servers 255

10.3 Eigenschaften der HIMax Steuerung 256

10.4 Erforderliche Aktionen bei Änderungen 257

10.5 Forcen von Globalen Variablen auf E/A Modulen 257

10.6 Konfiguration einer OPC-Server Verbindung 258

10.6.1 Benötigte Software: 258
10.6.2 Bedingungen für den X-OPC-Server Betrieb: 258
10.6.3 Installation auf dem Host-PC 259
10.6.4 OPC Server in SILworX konfigurieren 262
10.6.5 Einstellungen des OPC-Server im safeethernet Editor 263
10.6.6 X-OPC Data Access Server in SILworX konfigurieren 264
10.6.7 X-OPC Alarm & Event Server in SILworX konfigurieren 266
10.6.8 Parametrierung der Views und Prioritäten in SILworX 269

10.7 Alarm & Event Editor 272

10.7.1 Boolesche Ereignisse 272
10.7.2 Skalare Ereignisse 273

10.8 Parameter der X-OPC Server Eigenschaften 276

10.8.1 OPC-Server-Set 276
10.8.2 OPC-Server 280

10.9 Deinstallation des X-OPC-Servers 280

11 ComUserTask 281

11.1 Systemanforderung 281

11.1.1 Anlegen einer ComUserTask 281

11.2 Voraussetzungen 282

11.3 Abkürzungen 282

11.4 CUT-Schnittstelle in SILworX 283

11.4.1 Schedule-Intervall [ms] 283
11.4.2 Scheduling-Vorlauf 283
11.4.3 Scheduling-Nachlauf 283
11.4.4 STOP_INVALID_CONFIG 284
11.4.5 Variablen der CUT-Schnittstelle (CPU<->CUT) 284

11.5 CUT Funktionen 287

11.5.1 COM-User-Callback-Funktionen 287
11.5.2 COM-User-Library-Funktionen 287
11.5.3 Header-Files 287
11.5.4 Code-/Datenbereich und Stack für die CUT 288
11.5.5 Startfunktion CUCB_TaskLoop 288
11.5.6 Serielle Schnittstellen RS485 / RS232 IF 289
11.5.7 UDP/TCP-Socket-IF 297
11.5.8 Timer-IF 311
11.5.9 Diagnose 312

11.6 Installation der Entwicklungsumgebung ... 313
11.6.1 Installation der Cygwin-Umgebung ... 313
11.6.2 Installation des GNU Compilers ... 315
11.7 Neues CUT-Projekt anlegen ... 317
11.7.1 CUT-Makefiles ... 318
11.7.2 C-Quellcode bearbeiten ... 320
11.7.3 ComUserTask in das Projekt einbinden ... 324
11.7.4 Fehler beim Laden einer Konfiguration mit CUT ... 327
12 Allgemein ... 328
12.1 Konfiguration der Funktionsbausteine ... 328
12.1.1 Beschaffung der Funktionsbausteinbibliotheken ... 328
12.1.2 Konfiguration der Funktionsbausteine im Anwenderprogramm ... 328
12.1.3 Konfiguration der Funktionsbausteine im Strukturbaum von SILworX ... 329
12.2 Maximale Kommunikationszeitscheibe ... 331
Anhang ... 333
Glossar ... 333
Abbildungsverzeichnis ... 334
Tabellenverzeichnis ... 336
Index ... 342

1 Einleitung

Das Kommunikationshandbuch beschreibt die Eigenschaften und die Konfiguration der Kommunikationsprotokolle des sicherheitsgerichteten Steuerungssystems HIMax.

Mit den bereitgestellten Protokollen können Sie HIMax Steuerungen untereinander und mit Steuerungen anderer Hersteller verbinden.

Die Kenntnis von Vorschriften und das technisch einwandfreie Umsetzen der in diesem Handbuch enthaltenen Hinweise durch qualifiziertes Personal sind Voraussetzung für die Planung, Projektierung, Programmierung, Installation, Inbetriebnahme, Betrieb und Instandhaltung der HIMax Steuerungen.

Bei nicht qualifizierten Eingriffen in die Geräte, bei Abschalten oder Umgehen (Bypass) von Sicherheitsfunktionen oder bei Nichtbeachtung von Hinweisen dieses Handbuchs (und dadurch verursachten Störungen oder Beeinträchtigungen von Sicherheitsfunktionen) können schwere Personen-, Sach- oder Umweltschäden eintreten, für die HIMA keine Haftung übernehmen kann.

HIMax Automatisierungsgeräte werden unter Beachtung der einschlägigen Sicherheitsnormen entwickelt, gefertigt und geprüft. Nur für die in den Beschreibungen vorgesehenen Einsatzfälle mit den spezifizierten Umgebungsbedingungen verwenden.

1.1 Aufbau und Gebrauch des Handbuchs

Der Inhalt dieses Handbuchs ist Teil der Hardware-Beschreibung des programmierbaren elektronischen Systems HIMax.

Das Handbuch ist in folgende Hauptkapitel gegliedert:

- Einleitung
- Sicherheit
- Produktbeschreibung
- Inbetriebnahme
- Betrieb
- Instandhaltung
- Außerbetriebnahme
- Transport
- Entsorgung

Zusätzlich sind die folgenden Dokumente zu beachten:

Name	Inhalt	Dokumenten-Nr.
HIMax Systemhandbuch	Hardware-Beschreibung HIMax System	HI 801 000 D
HIMax Sicherheitshandbuch	Sicherheitsfunktionen des HIMax Systems	HI 801 002 D
HIMax Kommunikationshandbuch	Beschreibung der Kommunikation und Protokolle	HI 801 100 D
Erste Schritte	Einführung in SILworX	HI 801 102 D

Tabelle 1: Zusätzlich geltende Handbücher

Die aktuellen Handbücher finden Sie immer auf der HIMA Webseite www.hima.de. Anhand des Revisionsindex in der Fußzeile können Sie die Aktualität eventuell vorhandener Handbücher mit der Internetausgabe vergleichen.

1.2 Zielgruppe

Dieses Dokument wendet sich an Planer, Projekteure und Programmierer von Automatisierungsanlagen sowie Personen, die zu Inbetriebnahme, Betrieb und Wartung der Geräte und Systeme berechtigt sind. Vorausgesetzt werden spezielle Kenntnisse auf dem Gebiet der sicherheitsgerichteten Automatisierungssysteme.

1.3 Darstellungskonventionen

Zur besseren Lesbarkeit und zur Verdeutlichung gelten in diesem Dokument folgende Schreibweisen:

Fett	Hervorhebung wichtiger Textteile. Bezeichnungen von Schaltflächen, Menüpunkten und Registern in SILworX, auf die Sie klicken können
Kursiv	Parameter und Systemvariablen
`Courier`	Wörtliche Benutzereingaben
RUN	Bezeichnungen von Betriebszuständen in Großbuchstaben
Kap. 1.2.3	Querverweise sind Hyperlinks, auch wenn sie nicht besonders gekennzeichnet sind. Wenn Sie den Mauszeiger darauf positionieren, verändert er seine Gestalt. Bei einem Klick springt das Dokument zur betreffenden Stelle.

Sicherheits- und Gebrauchshinweise sind besonders gekennzeichnet.

1.3.1 Sicherheitshinweise

Die Sicherheitshinweise im Dokument sind wie folgend beschrieben dargestellt.
Um ein möglichst geringes Risiko zu gewährleisten, sind sie unbedingt zu befolgen. Der inhaltliche Aufbau ist

- Signalwort: Gefahr, Warnung, Vorsicht, Hinweis
- Art und Quelle der Gefahr
- Folgen der Gefahr
- Vermeidung der Gefahr

⚠ SIGNALWORT

Art und Quelle der Gefahr!
Folgen der Gefahr
Vermeidung der Gefahr

Die Bedeutung der Signalworte ist

- Gefahr: Bei Missachtung folgt schwere Körperverletzung bis Tod
- Warnung: Bei Missachtung droht schwere Körperverletzung bis Tod
- Vorsicht: Bei Missachtung droht leichte Körperverletzung
- Hinweis: Bei Missachtung droht Sachschaden

HINWEIS

Art und Quelle des Schadens!
Vermeidung des Schadens

1.3.2 Gebrauchshinweise

Zusatzinformationen sind nach folgendem Beispiel aufgebaut:

i	An dieser Stelle steht der Text der Zusatzinformation.

Nützliche Tipps und Tricks erscheinen in der Form:

TIPP	An dieser Stelle steht der Text des Tipps.

2 Sicherheit

Nachfolgende Sicherheitsinformationen, Hinweise und Anweisungen unbedingt lesen. Das Produkt nur unter Beachtung aller Richtlinien und Sicherheitsrichtlinien einsetzen.

Dieses Produkt wird mit SELV oder PELV betrieben. Von dem Modul selbst geht keine Gefahr aus. Einsatz im Ex-Bereich nur mit zusätzlichen Maßnahmen erlaubt.

2.1.1 Einsatzbedingungen

Die Geräte wurden für die Einhaltung der Anforderungen der folgenden Normen für EMV, Klima- und Umweltanforderungen entwickelt:

Norm	Inhalt
EC/EN 61131-2	Speicherprogrammierbare Steuerungen, Teil 2 Betriebsmittelanforderungen und Prüfungen
IEC/EN 61000-6-2	EMV Fachgrundnorm, Teil 6-2 Störfestigkeit, Industriebereich
IEC/EN 61000-6-4	Elektromagnetische Verträglichkeit (EMV) Fachgrundnorm Störaussendung, Industriebereich

Tabelle 2: Normen für EMV-, Klima- und Umweltanforderungen

Für den Einsatz der sicherheitsgerichteten Steuerungssysteme HIMax sind die nachfolgenden allgemeinen Bedingungen einzuhalten:

Art der Bedingung	Inhalt der Bedingung
Schutzklasse	Schutzklasse II nach IEC/EN 61131-2
Verschmutzung	Verschmutzungsgrad II nach IEC/EN 61131-2
Aufstellhöhe	< 2000 m
Gehäuse	Standard: IP20 Falls es die zutreffenden Applikationsnormen (z. B. EN 60204, EN 954-1) fordern, muss das Gerät in ein Gehäuse der geforderten Schutzart (z. B. IP 54) eingebaut werden.

Tabelle 3: Allgemeine Bedingungen

Klimatische Bedingungen

Die wichtigsten Prüfungen und Grenzwerte für klimatische Bedingungen sind in nachstehender Tabelle aufgelistet:

IEC/EN 61131-2	Klimaprüfungen
	Betriebstemperatur: 0...+60 °C (Prüfgrenzen: -10...+70 °C)
	Lagertemperatur: -40...+85 °C
	Trockene Wärme und Kälte; Beständigkeitsprüfungen: +70 °C / -25 °C, 96 h, Stromversorgung nicht angeschlossen
	Temperaturwechsel; Beständigkeits- und Unempfindlichkeitsprüfung: -25 °C / +70 °C und 0 °C / +55 °C, Stromversorgung nicht angeschlossen
	Zyklen mit feuchter Wärme; Beständigkeitsprüfungen: +25 °C / +55 °C, 95 % relative Feuchte, Stromversorgung nicht angeschlossen

Tabelle 4: Klimatische Bedingungen

Mechanische Bedingungen

Die wichtigsten Prüfungen und Grenzwerte für mechanische Bedingungen sind in nachstehender Tabelle aufgelistet:

IEC/EN 61131-2	Mechanische Prüfungen
	Unempfindlichkeitsprüfung gegen Schwingungen: 5...9 Hz / 3,5 mm 9...150 Hz, 1 g, Prüfling in Betrieb, 10 Zyklen pro Achse
	Unempfindlichkeitsprüfung gegen Schocken: 15 g, 11 ms, Prüfling in Betrieb, 2 Zyklen pro Achse

Tabelle 5: Mechanische Prüfungen

EMV-Bedingungen

Für sicherheitsbezogene Systeme werden erhöhte Pegel bei der Störbeeinflussung gefordert. HIMax Systeme erfüllen diese Anforderungen nach IEC 62061 und IEC 61326-3-1 (DIS). Siehe die Spalte „Kriterium FS“ (Funktionale Sicherheit).

IEC/EN 61131-2	Prüfungen der Störfestigkeit	Kriterium FS
IEC/EN 61000-4-2	ESD-Prüfung: 6 kV Kontakt-, 8 kV Luftentladung	-
IEC/EN 61000-4-3	RFI-Prüfung (10 V/m): 26 MHz...1 GHz, 80 % AM RFI-Prüfung (20 V/m): 26 MHz...2,7 GHz, 80 % AM: EN 298	- 20 V/M
IEC/EN 61000-4-4	Burst-Prüfung: 2 kV Versorgungs-, 1 kV Signalleitungen	4 kV
IEC/EN 61000-4-12	Prüfung mit gedämpften Schwingungen: 2,5 kV L-,L+ / PE 1 kV L+ / L -	

Tabelle 6: Prüfungen der Störfestigkeit

IEC/EN 61000-6-2	Prüfungen der Störfestigkeit	Kriterium FS
IEC/EN 61000-4-6	Hochfrequenz, asymmetrisch: 10 V, 150 kHz...80 MHz, AM 20 V, 150 kHz...80 MHz, AM: EN 298	 20 V
IEC/EN 61000-4-3	900 MHz-Impulse	
IEC/EN 61000-4-5	Stoßspannung: 2 kV, 1 kV	2 kV / 1 kV

Tabelle 7: Prüfungen der Störfestigkeit

IEC/EN 61000-6-4	Prüfungen der Störaussendung
EN 55011 Klasse A	Störaussendung: gestrahlt, leitungsgebunden

Tabelle 8: Prüfungen der Störaussendung

Spannungsversorgung

Die wichtigsten Prüfungen und Grenzwerte für die Spannungsversorgung der Geräte sind in nachstehender Tabelle aufgelistet:

IEC/EN 61131-2	Nachprüfung der Eigenschaften der Gleichstromversorgung
	Die Spannungsversorgung muss alternativ folgende Normen erfüllen: IEC/EN 61131-2 oder SELV (Safety Extra Low Voltage) oder PELV (Protective Extra Low Voltage)
	Die Absicherung der HIMax-Geräte muss gemäß den Angaben dieses Handbuchs erfolgen
	Prüfung des Spannungsbereiches: 24 VDC, -20 %...+25 % (19,2 V...30,0 V)
	Prüfung auf Unempfindlichkeit gegen Kurzzeitunterbrechung der externen Stromversorgung: DC, PS 2: 10 ms
	Polaritätsumkehr der Versorgungsspannung: Hinweis im entsprechenden Kapitel des Systemhandbuchs oder im Datenblatt der Stromversorgung.
	Pufferdauer, Beständigkeitsprüfung: Prüfung B, 1000 h

Tabelle 9: Nachprüfung der Eigenschaften der Gleichstromversorgung

ESD-Schutzmaßnahmen

Nur Personal, das Kenntnisse über ESD-Schutzmaßnahmen besitzt, darf Änderungen oder Erweiterungen des Systems oder den Austausch eines Moduls durchführen.

HINWEIS

Elektrostatische Entladungen können die in den Steuerungen eingebauten elektronischen Bauteile beschädigen!

- Benutzen Sie für die Arbeiten einen antistatisch gesicherten Arbeitsplatz und tragen Sie ein Erdungsband.
- Bewahren Sie Module bei Nichtbenutzung elektrostatisch geschützt auf, z. B. in der Verpackung.

Änderungen oder Erweiterungen an der Verdrahtung des Systems darf nur durch Personal durchgeführt werden, das Kenntnis von ESD-Schutzmaßnahmen besitzt.

2.2 Restgefahren

Von einem HIMax Modul selbst geht keine Gefahr aus.

Restgefahren können ausgehen von:

- Fehlern in der Projektierung
- Fehlern im Anwenderprogramm
- Fehlern in der Verdrahtung

2.3 Sicherheitsvorkehrungen

Am Einsatzort geltende Sicherheitsbestimmungen beachten und vorgeschriebene Schutzausrüstung tragen. Für die Montage des Basisträgers X-BASE PLATE ist das Tragen von Sicherheitsschuhen vorgeschrieben.

2.4 Notfallinformation

Eine HIMax Steuerung ist Teil der Sicherheitstechnik einer Anlage. Der Ausfall einer Steuerung bringt die Anlage in den sicheren Zustand.

Im Notfall ist jeglicher Eingriff in die Sicherheitstechnik verboten.

3 Produktbeschreibung

Das System HIMax ist ein sicherheitsgerichtetes Steuerungssystem, das für Dauerbetrieb und maximale Verfügbarkeit ausgelegt ist.

HIMax ist ein modulares System. Die Funktionen der Verarbeitung, Ein- und Ausgabe, und Kommunikation sind auf steckbare Module verteilt.

Neben den physikalischen Eingangs- und Ausgangsvariablen des HIMax Systems können Variablen auch über eine Datenverbindung mit einem anderen System ausgetauscht werden.

Für den Prozessdatenaustausch stehen das sicherheitsgerichtete Protokoll safe**ethernet** und verschiedene Standardprotokolle bereit.

3.1 Sicherheitsgerichtetes Protokoll (safeethernet)

Alle HIMax Systeme können sicherheitsgerichtet in SIL 3 über Ethernet kommunizieren. Die sicherheitsgerichtete Kommunikation erfolgt über das safe**ethernet** Protokoll.

Das sicherheitsgerichtete safe**ethernet** Protokoll wird zum sicherheitsgerichteten Prozessdatenaustausch zwischen HIMax und HIMatrix Steuerungen in einem Ethernet-Netzwerk verwendet werden.

Prozessormodule:	safe**ethernet** wird auf dem CPU-Modul ausgeführt. Maximal 4 CPU-Module pro HIMax Steuerung.
Verbindungen:	pro HIMax: maximal 255 safe**ethernet** Verbindungen pro HIMatrix: maximal 64 safe**ethernet** Verbindungen
Transportwege:	Ethernet-Schnittstellen der CPU- und COM-Module. Verwendete Ethernet-Schnittstellen simultan auch für andere Protokolle nutzbar.
Prozessdatenmenge:	Über jede safe**ethernet** Verbindung können je Richtung maximal 1100 Bytes übertragen werden (HIMatrix max. 900 Bytes).

i Die maximale Prozessdatenmenge der Steuerung von 512 kByte darf nicht überschritten werden. In diesem Fall wird die Parametrierung der Steuerung beim Laden abgelehnt.

Beschreibung des sicherheitsgerichteten Protokolls safe**ethernet** ab Kapitel 4.

⚠ WARNUNG

Manipulation der sicherheitsgerichteten Datenübertragung!

Personenschaden

Der Betreiber hat dafür zu sorgen, dass das für safeethernet verwendete Ethernet ausreichend vor Manipulationen (z. B. durch Hacker) geschützt wird.
Art und Umfang der Maßnahmen sind mit der abnehmenden Prüfstelle abzustimmen.

3.2 Standardprotokolle

Für eine optimale Integration des HIMax Systems mit Feldgeräten und Leitsystemen stehen zusätzliche bewährte Standardprotokolle zur Verfügung. Hierfür finden sowohl Ethernet als auch Feldbusprotokolle Verwendung.

Eine Reihe von Kommunikationsprotokollen erlaubt nur eine nicht sicherheitsgerichtete Übertragung von Daten. Diese sind nur für nicht sicherheitsgerichtete Teile einer Automatisierungsaufgabe verwendbar.

Kommunikationsmodule:	Standardprotokolle werden auf dem COM-Modul ausgeführt
Transportwege:	Ethernet-Schnittstellen und Feldbusschnittstellen der COM-Module.
Maximale Anzahl Standardprotokolle	HIMax: ▪ 20 COM-Module pro HIMax Steuerung. ▪ 20 Standardprotokolle pro HIMax Steuerung. ▪ 6[1)] Standardprotokolle pro COM-Modul HIMatrix: ▪ 4 Standardprotokolle pro HIMatrix Steuerung.
Prozessdatenmenge:	Mit allen nichtsicherheitsgerichteten Protokollen können pro HIMax Steuerung insgesamt 128 kB Daten gesendet und 128 kB Daten empfangen werden. pro HIMatrix Steuerung insgesamt 64 kB Daten gesendet und 64 kB Daten empfangen werden.

1) SNTP-Client und SNTP-Server gehen in diese Rechnung nicht mit ein

⚠ WARNUNG

Verwendung von unsicheren Importdaten

Personenschaden

Nichtsichere Daten dürfen nicht für die Sicherheitsfunktionen des Anwenderprogramms verwendet werden.

Die folgenden Standardprotokolle stehen zur Verfügung:

Protokoll	pro Modul	Beschreibung
PROFINET I/O Controller	1	Kapitel 5.2
PROFINET I/O Device	1	Kapitel 5.7
PROFIBUS DP Master	2	Kapitel 6.1
PROFIBUS DP Slave	1	Kapitel 6.13
Modbus Master	1	Kapitel 7.1
Modbus Slave	1	Kapitel 7.9
S&R TCP	1	Kapitel 8
HIMA X-OPC Server[1)]	---	Kapitel 10
ComUserTask	1	Kapitel 11

Tabelle 10: Verfügbare Standardprotokolle

1) Der HIMA X-OPC Server wird auf einem Host-PC installiert und dient als Übertragungsschnittstelle zwischen bis zu 255 HIMax Steuerungen und Fremdsystemen, die über eine OPC Schnittstelle verfügen.

i Maximal 64 TCP-Verbindungen pro HIMatrix Steuerung oder HIMax COM-Modul.
Maximal 1280 TCP-Verbindungen pro HIMax Steuerung mit 20 COM-Modulen.

Maximale Anzahl aktivierter Protokolle auf einer HIMatrix oder einem HIMax COM-Modul

Maximal 64 TCP Sockets sind pro HIMatrix oder HIMax COM-Modul verfügbar.

Beispiel 1:

Protokoll	Verbindungen
1 Modbus Master	TCP 44 Slave-Verbindungen, RTU 122 Slave-Verbindungen
1 Modbus Slave	TCP 20 Master-Verbindungen, RTU 1 Master-Verbindung

Tabelle 11:Protokolle auf einem Kommunikationsmodul

Beispiel 2:

Protokoll	Verbindungen
1 PROFIBUS DP Master	122 Slave-Verbindungen
1 PROFIBUS DP Slave	1 Master-Verbindung

Tabelle 12:Protokolle auf einem Kommunikationsmodul

3.3 Redundanz

Das HIMax System ist als hochverfügbares System konzipiert und stellt auch für die Kommunikation Redundanz bereit. Eine Kommunikationsverbindung ist redundant, wenn zwei physikalische Transportwege existieren.

Für HIMatrix ist eine redundante Kommunikation nur über safe**ethernet** vorgesehen.

Redundanz über safeethernet

Die Redundanz konfigurieren Sie im safe**ethernet**-Editor, indem Sie die Ethernetschnittstellen für den zweiten Transportweg auswählen (siehe Kapitel 4.2).

Redundanz über Standardprotokolle

PROFIBUS DP Master PROFIBUS DP Slave PROFINET IO TCP S&R Modbus Master	Die Redundanz des Standardprotokolls müssen Sie im Anwenderprogramm so konfigurieren, dass dieses die redundanten Transportwege überwacht und die redundant übertragenen Prozessdaten dem jeweiligen Transportweg zuordnet.
Modbus Slave	Redundanz kann in SILworX eingestellt werden.

3.4 Aufbau der HIMax COM-Modul Teilenummer

Das X-COM 01 bildet mit dem Connector Board X-CB 001 02 eine funktionale Einheit. Das Connector Board muss separat bestellt werden.

Bei der Ausrüstung der X-COM 01 mit einem oder mehreren Fieldbus Submodulen ändert sich die Bezeichnung des Moduls von X-COM 01 nach X-COM 010 XY* siehe Tabelle 13.

Nachfolgend finden Sie die verfügbaren Komponenten und deren Teilenummern.

Bezeichnung	Beschreibung	Teile-Nr.
X-COM 01	Kommunikationsmodul ohne Fieldbus Submodul	98 5260000
X-COM 010 **XY**	Kommunikationsmodul mit Fieldbus Submodul	98 52600**XY**
X-CB 001 02	Connector Board	98 5020008

Tabelle 13: Teilenummern

X: Option für Feldbus Submodule 1 (Feldbusschnittstelle 1)

Y: Option für Feldbus Submodule 2 (Feldbusschnittstelle 2)

Optionswerte für **X** und **Y**:

0: kein Fieldbus Submodul eingebaut

1: RS485 (Modbus Master, Modbus Slave wird durch Lizenzcode festgelegt) oder CUT (ComUserTask)

2: PROFIBUS DP Master

3: PROFIBUS DP Slave

4 ---

5 Submodul RS232 für CUT (ComUserTask) eingebaut

6 Submodul RS422 für CUT (ComUserTask) eingebaut

Die Teilenummer (Part-Nr.) ist auf dem Typenschild des Moduls abgedruckt.

Beispiele:

Teilenummer	Feldbus Submodule 1 (FB1)	Feldbus Submodule 2 (FB2)
98 52600 21	PROFIBUS Master	RS485
98 52600 23	PROFIBUS Master (max. 12 MBit)	PROFIBUS Slave (max. 1,5 MBit)
98 52600 11	RS485	RS485
98 52600 00	-	-

Tabelle 14: Beispiele für Teilenummern des COM-Moduls

i HIMA empfiehlt, PROFIBUS DP über die Feldbusschnittstelle FB1 (Übertragungsrate maximal 12 MBit) zu betreiben. Über die Feldbusschnittstelle FB2 ist eine maximale Übertragungsrate von 1,5 MBit zugelassen.

⚠ VORSICHT

Unsachgemäßes Öffnen des COM-Moduls

Schaden am COM-Modul

Die Bestückung der Kommunikationsmodule mit Feldbus Submodulen erfolgt ausschließlich durch HIMA.

3.5 Registrierung und Aktivierung der Protokolle

Die folgenden Protokolle sind für die HIMax verfügbar und können wie folgt aktiviert werden:

Protokoll	Schnittstellen	Aktivierung
HIMA safe**ethernet**	Ethernet	[1]
HIMA X-OPC Server (läuft auf Host-PC)	Ethernet	[4]
PROFINET IO Controller	Ethernet	[4]
PROFINET IO Device	Ethernet	[4]
Modbus TCP Master	Ethernet	[4]
Modbus TCP Slave	Ethernet	[4]
TCP Send/Receive	Ethernet	[4]
SNTP Server/Client	Ethernet	[4]
PROFIBUS DP Master	FB1 und FB2	[2]
PROFIBUS DP Slave	FB1 oder FB2	[2]
Modbus Master RS485	FB1 oder FB2	[3]
Modbus Slave RS485	FB1 oder FB2	[3]
CUT (ComUserTask) RS232 und RS422	FB1 oder FB2	[3]

Tabelle 15:Verfügbare Protokolle der HIMax

[1]. safe**ethernet** ist bei allen HIMax Systemen standardmäßig freigeschaltet.

[2]. Für PROFIBUS Master und PROFIBUS Slave erfolgt die Freischaltung durch Einbau eines Fieldbus-Submoduls.

[3]. Für Fieldbus-Submodul RS485 (Modbus RS485) und Fieldbus-Submodul RS232, RS422 (ComUserTask) müssen Sie zusätzlich einen Software-Freischaltcode für Ihr gewähltes Feldbusprotokoll erwerben.

[4]. Den Software-Freischaltcode generieren Sie auf der HIMA Webseite mit der System-ID (z.B. 60000) Ihrer Steuerung, folgen Sie den Anweisungen auf der HIMA Webseite.

i Der Software-Freischaltcode ist untrennbar mit dieser System ID verbunden. Eine Lizenz kann nur einmalig für eine bestimmte System ID genutzt werden. Deshalb sollten Sie die Freischaltung erst durchführen, wenn die System ID eindeutig feststeht.

Sie können alle Ethernetprotokolle ohne Software-Freischaltcode für 5000 Betriebsstunden testen.

i Generieren Sie rechtzeitig Ihren Software-Freischaltcode!
Nach Ablauf der 5000 Betriebsstunden läuft die Kommunikation weiter bis die Steuerung gestoppt wird. Danach lässt sich das Anwenderprogramm ohne gültigen Software-Freischaltcode für die projektierten Protokolle nicht mehr starten (ungültige Konfiguration).

So tragen Sie den Software-Freischaltcode in SILworX ein:

1. Wählen Sie im Strukturbaum **Configuration, Ressource, Lizenzverwaltung**.
2. Rechtsklick auf **Lizenzverwaltung** und im Kontextmenü **Neu, Lizenzschlüssel** wählen.
   ☑ Der Lizenzschlüssel wird neu hinzugefügt.
3. Rechtsklick auf **Lizenzschlüssel** und im Kontextmenü **Eigenschaften** wählen.
4. Tragen Sie im Feld **Freischaltcode** den generierten Software-Freischaltcode ein.

3.6 Ethernet-Schnittstellen

Die Kommunikation mit externen Systemen erfolgt über die Ethernet-Schnittstellen der CPU- und COM Module. Jede Ethernet-Schnittstelle kann simultan mehrere Protokolle verarbeiten.

i Über die Ethernet-Schnittstellen der Systembusbaugruppe X-SB 01 ist keine Prozessdatenkommunikation möglich. Die Ethernet-Schnittstellen UP und DOWN sind ausschließlich für die Verbindung der HIMax Basisträger untereinander vorgesehen.

3.6.1 Eigenschaften der Ethernet-Schnittstellen

Eigenschaft	CPU Modul	HIMax COM Modul HIMatrix
Ports	4	4
Übertragungsstandard	10/100/1000 Base-T, Halb- und Vollduplex	10/100 Base-T, Halb- und Vollduplex
Auto Negotiation	Ja	Ja
Auto-Crossover	Ja	Ja
Anschlussbuchse	RJ 45	RJ 45
IP-Adresse	Frei Konfigurierbar[1)]	Frei Konfigurierbar[1)]
Subnetz-Maske	Frei Konfigurierbar[1)]	Frei Konfigurierbar[1)]
Unterstützte Protokolle	safe**ethernet** Programmiergerät (PADT), SNTP	safe**ethernet** Standardprotokolle

Tabelle 16: Eigenschaften Ethernet Schnittstellen

[1)] Allgemein gültige Regeln für die Vergabe von IP-Adressen und Subnetz-Masken müssen beachtet werden.

Jedes COM- und CPU-Modul besitzt einen Ethernet-Switch mit einer IP-Adresse und vier Ports auf dem zugehörigen Connector Board.

Der Ethernet-Switch eines CPU- oder COM Moduls baut eine gezielte Verbindung zwischen zwei Kommunikationspartnern für die Übertragung von Daten auf. Das verhindert Kollisionen und entlastet das Netzwerk.

Zur gezielten Weiterleitung der Daten wird eine MAC-/IP Adressen Zuordnungstabelle (ARP-Cache) angelegt und MAC-Adressen bestimmten IP-Adressen zugeordnet. Datenpakete werden jetzt nur noch an die IP-Adressen weitergeleitet, die im ARP-Cache gelistet sind.

i Austausch eines CPU/COM Moduls mit gleicher IP-Adresse
Wird ein CPU/COM Modul ausgetauscht für welches *ARP Aging time* = 5 Minuten und *MAC-Learning* = Konservativ eingestellt wurde, so übernimmt der Kommunikationspartner erst nach minimal 5 Minuten bis maximal 10 Minuten die neue MAC-Adresse. In dieser Zeit ist keine Kommunikation über das getauschte CPU/COM Modul möglich.

Neben der einstellbaren ARP Aging time muss mindestens die nicht änderbare MAC Aging-Time des Switch (ca. 10 Sekunden) abgewartet werden, bis wieder eine Kommunikation über das getauschte CPU/COM Modul möglich ist.

3.6.2 Konfiguration der Ethernet-Schnittstellen

Die Konfiguration der Ethernet-Schnittstelle erfolgt über die Detailansicht des CPU- oder COM Moduls.

Für HIMax Steuerungen sind die Standardwerte der Parameter *Speed Modus* und *Flow-Control Modus* auf AutoNeg eingestellt.

i Kommunikationsverlust!

Bei einer ungünstigen Einstellung der Ethernet-Parameter ist das Prozessor- oder Kommunikationsmodul nicht mehr erreichbar. Führen Sie einen Reset des Moduls durch!

So öffnen Sie die Detailansicht des Kommunikationsmoduls:

1. Wählen Sie im Strukturbaum **Konfiguration, Ressource, Hardware**.
2. Rechtsklick auf **Hardware** und im Kontextmenü **Edit** wählen, um den Hardware Editor zu öffnen.
3. Rechtsklick auf das **Kommunikationsmodul** und im Kontextmenü **Detailansicht** wählen, um die Detailansicht zu öffnen.

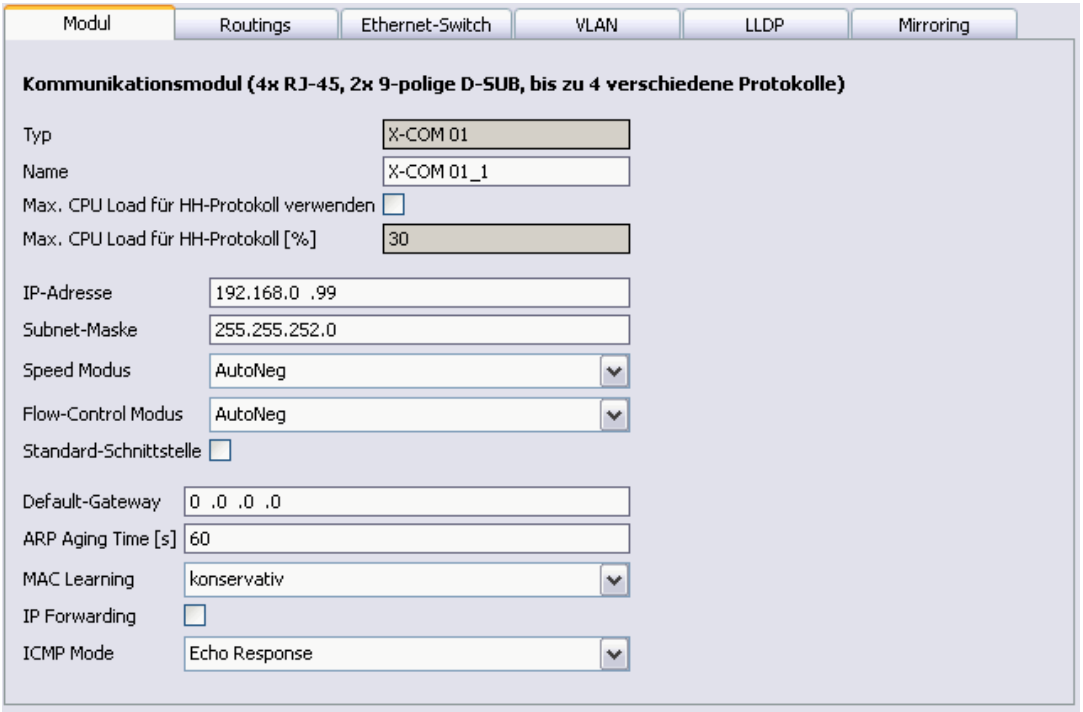


Bild 1: Konfigurationsdialog für CPU- und COM Modul in SILworX

i Die Einträge in den Eigenschaften der COM- und CPU-Module müssen Sie mit dem Anwenderprogramm neu kompilieren und in die Steuerung übertragen, damit sie für die Kommunikation des HIMax Systems wirksam werden.

Modul

Element	Beschreibung
Name	Name des Kommunikationsmoduls.
Max. CPU Load für HH-Protokoll verwenden	▪ Aktiviert: Limit der CPU-Last aus dem Feld *Max. CPU Load [%]* übernehmen. ▪ Deaktiviert: Kein Limit der CPU-Last, für safe**ethernet** verwenden.
Max. CPU Load für HH-Protokoll [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des safe**ethernet** Protokolls produziert werden darf. i Die Maximale Last muss unter allen verwendeten Protokollen aufgeteilt werden, welche dieses Kommunikationsmodul benutzen.
IP-Adresse	IP-Adresse der Ethernet-Schnittstelle
Subnet-Maske	32-Bit-Adressmaske zur Unterteilung einer IP-Adresse in Netzwerk- und Host-Adresse.
Speed-Modus Flow-Control-Modus	i Nur Einstellung Autoneg erlaubt! Bei anderen Einstellungen geht das Modul in STOPP.
Standard-Schnittstelle	Aktiviert: Schnittstelle wird als Standardschnittstelle für den System-Login verwendet.
Default-Gateway	IP-Adresse des Default Gateway
Erweiterte Funktionen nutzen	Die Parameter *ARP Aging Time [s]*, *MAC Learning* und *IP Forwarding* nutzen.
ARP Aging Time [s]	Ein CPU- oder COM Modul speichert die MAC-Adressen seiner Kommunikationspartner in einer MAC-/IP Adresse Zuordnungstabelle (ARP-Cache). Wenn während einer Zeitspanne von 1 bis 2 mal *ARP Aging Time* - Nachrichten vom Kommunikationspartner eintreffen, bleibt die MAC-Adresse im ARP-Cache erhalten. - keine Nachrichten vom Kommunikationspartner eintreffen, wird die MAC-Adresse aus dem ARP-Cache gelöscht. Der typische Wert für die *ARP Aging Time* in einem lokalen Netzwerk ist 5 s ... 300 s. Der Inhalt des ARP-Cache kann vom Anwender nicht ausgelesen werden. Bei der Verwendung von Routern oder Gateways *ARP Aging Time* an die zusätzlichen Verzögerungen für Hin- und Rückweg anpassen (erhöhen). Bei zu geringer *ARP Aging Time* löscht das CPU-/COM Modul die MAC-Adresse des Kommunikationspartners aus dem ARP-Cache und die Kommunikation wird nur verzögert ausgeführt oder bricht ab. Für einen effizienten Einsatz muss die ARP Aging Time > der ReceiveTimeouts der verwendeten Protokolle sein. Wertebereich: 1 s ... 3600 s Standardwert: 60 s

MAC Learning	Mit MAC Learning und *ARP Aging Time* stellt der Anwender ein, wie schnell eine MAC-Adresse gelernt werden soll. Folgende Einstellungen sind möglich: ▪ Konservativ (Empfohlen): Wenn sich im ARP-Cache bereits MAC-Adressen von Kommunikationspartnern befinden, so sind diese Einträge für die Dauer von mindestens 1 mal *ARP Aging Time* bis maximal 2 mal *ARP Aging Time* verriegelt und können nicht durch andere MAC-Adressen ersetzt werden. Dadurch ist sichergestellt, dass Datenpakete nicht absichtlich oder unabsichtlich auf fremde Netzwerkteilnehmer umgeleitet werden können (ARP spoofing). ▪ Tolerant: Beim Empfang einer Nachricht wird die IP-Adresse in der Nachricht mit den Daten im ARP-Cache verglichen und die gespeicherte MAC-Adresse im ARP-Cache sofort mit der MAC-Adresse aus der Nachricht überschrieben. Verwenden Sie die Einstellung Tolerant, wenn die Verfügbarkeit der Kommunikation wichtiger ist als der sichere Zugriff (authorized access) auf die Steuerung.
IP Forwarding	Ermöglicht einem CPU- oder COM Modul, als Router zu arbeiten und Datenpakete anderer Netzwerkknoten weiterzuleiten. ▪ Aktiviert: Weiterleitung ist eingeschaltet. ▪ Deaktiviert: Weiterleitung ist ausgeschaltet.
ICMP Mode	Das Internet Control Message Protocol (ICMP) ermöglicht den höheren Protokollschichten, Fehlerzustände auf der Vermittlungsschicht zu erkennen und die Übertragung der Datenpakete zu optimieren. Meldungstypen des Internet Control Message Protocol (ICMP), die von dem CPU-Modul unterstützt werden: ▪ keine ICMP-Antworten Alle ICMP-Befehle sind abgeschaltet. Dadurch wird eine hohe Sicherheit gegen Sabotage erreicht, die über das Netzwerk erfolgen könnte. ▪ Echo Response Wenn Echo Response eingeschaltet ist, antwortet der Knoten auf einen Ping-Befehl. Es ist somit feststellbar, ob ein Knoten erreichbar ist. Die Sicherheit ist immer noch hoch. ▪ Host unerreichbar Für den Anwender nicht von Bedeutung. Nur für Tests beim Hersteller. ▪ alle implementierten ICMP-Antworten Alle ICMP-Befehle sind eingeschaltet. Dadurch wird eine genauere Fehlerdiagnose bei Netzwerkstörungen erreicht.

Tabelle 17: Konfigurationsparameter

Routings

Element	Beschreibung
Name	Bezeichnung der Routing-Einstellung
IP Adresse	Ziel IP-Adresse des Kommunikationspartners (bei direktem Host-Routing) oder Netzwerkadresse (bei Subnet-Routing) Wertebereich: 0.0.0.0 ... 255.255.255.255 Standardwert: 0.0.0.0
Subnet Mask	Definiert Ziel-Adressbereich für einen Routing-Eintrag. 255.255.255.255 (bei direktem Host-Routing) oder Subnet-Maske des adressierten Subnetzes. Wertebereich: 0.0.0.0 ... 255.255.255.255 Standardwert: 255.255.255.255
Gateway	IP-Adresse des Gateways zum adressierten Netzwerk. Wertebereich: 0.0.0.0 ... 255.255.255.255 Standardwert: 0.0.0.1

Tabelle 18: Routing Parameter

Ethernet-Switch

Element	Beschreibung
Port	Nummer des Ports wie Gehäuseaufdruck; pro Port darf nur eine Konfiguration vorhanden sein. Wertebereich: 1 ... 4
Speed [Mbit/s]	10 MBit/s: Datenrate 10 MBit/s 100 MBit/s: Datenrate 100 MBit/s 1000 MBit/s: Datenrate 1000 MBit/s (CPU-Modul) Autoneg (10/100/1000): automatische Einstellung der Baudrate Standardwert: Autoneg
Flow-Control	Vollduplex: Kommunikation in beide Richtungen gleichzeitig Halbduplex: Kommunikation in eine Richtung Autoneg: automatische Kommunikationssteuerung Standardwert: Autoneg
Autoneg auch bei festen Werten	Das "Advertising" (Übermitteln der Speed und Flow-Control Eigenschaften) wird auch bei fest eingestellten Werten von *Speed* und *Flow-Control* durchgeführt. Hierdurch erkennen andere Geräte, deren Ports auf *Autoneg* eingestellt sind, die Einstellung der HIMax Ports.
Limit	Eingehende Multicast- und/oder Broadcast-Pakete limitieren. Aus: keine Limitierung Broadcast: Broadcast limitieren (128 kbit/s) Multicast und Broadcast: Multicast und Broadcast limitieren (1024 kbit/s) Standardwert: Broadcast

Tabelle 19: Ethernet-Switch-Parameter

VLAN (Port based VLAN)

Konfiguriert die Verwendung von port-based VLAN.

i Soll VLAN unterstützt werden, muss Port based VLAN abgeschaltet sein, so daß jeder Port mit jedem anderen Port des Switches kommunizieren kann.

Für jeden Port eines Switches können Sie einstellen, zu welchem anderen Port des Switches, empfangene Ethernet Frames gesendet werden dürfen.
Die Tabelle im Register VLAN enthält Einträge, mit denen die Verbindung zwischen zwei Ports aktiv oder inaktiv geschaltet werden kann.

	Eth1	Eth2	Eth3	Eth4
Eth1				
Eth2	aktiv			
Eth3	aktiv	aktiv		
Eth4	aktiv	aktiv	aktiv	
COM	aktiv	aktiv	aktiv	aktiv

Tabelle 20:Register VLAN

Standardeinstellung: alle Verbindungen zwischen den Ports aktiv

LLDP

LLDP (Link Layer Discovery Protocol) sendet per Multicast in periodischen Abständen Informationen über das eigene Gerät (z.B. MAC-Adresse, Gerätenamen, Portnummer) und empfängt die gleichen Informationen von Nachbargeräten.
Das Prozessor- und das Kommunikationsmodul unterstützen LLDP auf den Ports Eth1, Eth2, Eth3 und Eth4.

Die folgenden Parameter legen fest, wie der betreffende Port arbeitet:

Aus	LLDP ist auf diesem Port deaktiviert
Send	LLDP sendet LLDP Ethernet Frames, empfangene LLDP Ethernet frames werden gelöscht ohne diese zu verarbeiten
Receive	LLDP sendet keine LLDP Ethernet Frames, aber empfangene LLDP Frames werden verarbeitet
Send/Receive	LLDP sendet und verarbeitet empfangene LLDP Ethernet Frames

Standardeinstellung: Send/Receive

Mirroring

Konfiguriert, ob das Modul Ethernet-Pakete auf einen Port dupliziert, so dass sie von einem dort angeschlossenen Gerät mitgelesen werden können, z.B. zu Testzwecken.

Die folgenden Parameter legen fest, wie der betreffende Port arbeitet:

Aus	Dieser Port nimmt am Mirroring nicht teil.
Egress:	Ausgehende Daten dieses Ports werden dupliziert.
Ingress:	Eingehende Daten dieses Ports werden dupliziert
Egress/Ingress:	Ein- und ausgehende Daten dieses Ports .werden dupliziert.
Dest Port:	Duplizierte Daten werden auf diesen Port geschickt.

Standardeinstellung: Aus

3.6.3 Verwendete Netzwerkports für Ethernet-Kommunikation

UDP-Ports / Verwendung

8000	Programmierung und Bedienung mit SILworX
8001	Konfiguration der Remote I/O durch die PES
6010	safeethernet und OPC
123	SNTP (Zeitsynchronisation zwischen PES und Remote I/O, sowie externen Geräten)
8895	Modbus-Master-UDP falls konfiguriert
502	Modbus Slave (vom Anwender änderbar)

TCP Ports / Verwendung

8895	Modbus Master
502	Modbus Slave (vom Anwender änderbar)
Xxx	TCP-SR durch Anwender vergeben

3.7 Feldbusschnittstellen

Die Kommunikation mit externen Systemen kann über die Feldbusschnittstellen des COM-Moduls erfolgen. Über jede Feldbusschnittstelle ist jeweils nur ein Protokoll möglich.

Die Feldbusschnittstellen müssen mit einem Feldbus Submodul ausgestattet werden. Ohne Feldbus Submodul ist keine Kommunikation über diese Schnittstellen möglich. Das Feldbus Submodul legt den Übertragungsstandard der Schnittstelle fest.

Bezeichnung	Schnittstelle	Protokoll
FB 1	D-Sub-Buchse 9-polig	PROFIBUS Master PROFIBUS Slave Modbus Master RS485 Modbus Slave RS485 ComUserTask (RS 232)
FB 2	D-Sub-Buchse 9-polig	PROFIBUS Master PROFIBUS Slave Modbus Master RS485 Modbus Slave RS485 ComUserTask (RS 232)

Tabelle 21: Feldbusschnittstellen

3.7.1 Pin-Belegung der D-SUB-Buchsen FB1 und FB2

i Die PIN-Belegung der Feldbusschnittstellen ist vom verwendeten Feldbus Submodul abhängig und ist in den nachfolgenden Tabellen aufgeführt.

Feldbus Submodul für PROFIBUS DP Master oder -Slave

Pin	Signal	Funktion
1	---	---
2	---	---
3	RxD/TxD-A	Empfang/Sende-Daten A
4	RTS	Steuersignal
5	DGND	Daten-Bezugspotenzial
6	VP	+5 V Versorgungsspannung
7	---	---
8	RxD/TxD-B	Empfang/Sende-Daten B
9	---	---

Tabelle 22: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für PROFIBUS DP

Feldbus Submodul RS485 für Modbus Master oder -Slave

Pin	Signal	Funktion
1	---	---
2	RP	+5 V mit Dioden entkoppelt
3	RxD/TxD-A	Empfang/Sende-Daten A
4	CNTR-A	Steuersignal A
5	DGND	Daten-Bezugspotenzial
6	VP	+5 V Versorgungsspannung
7	---	---
8	RxD/TxD-B	Empfang/Sende-Daten B
9	CNTR-B	Steuersignal B

Tabelle 23: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für Modbus

Feldbus Submodul RS232 für ComUserTask

Pin	Signal	Funktion
1	---	---
2	TxD	Sende-Daten
3	RxD	Empfangs Daten
4	---	---
5	DGND	Daten-Bezugspotenzial
6	---	---
7	RTS	Anforderung zum Senden
8	---	---
9	---	---

Tabelle 24: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für RS232

4 safeethernet

Alle HIMax Systeme sind safe**ethernet** fähig. Sie können sicherheitsgerichtet in SIL 3 über Ethernet (HIMax 1 Gbit/s , HIMatrix 100 Mbit/s) kommunizieren.

Die jeweiligen Ethernet-Schnittstellen der HIMax Steuerungen sind simultan auch für andere Protokolle nutzbar.

Die safe**ethernet** Kommunikation zwischen den Steuerungen kann über verschiedene Ethernet-Netzwerktopologien erfolgen. Passen Sie die Parameter des safe**ethernet** Protokolls an das verwendete Ethernet-Netzwerk an, um Geschwindigkeit und Effizienz des Datentransfers zu erhöhen.

Diese Parameter können mit Hilfe so genannter Netzwerkprofile eingestellt werden. Die werkseitige Einstellung der Parameter stellt die Kommunikation sicher, ohne dass sich der Anwender zunächst in Details der Netzwerkkonfiguration einarbeiten muss.

i Das safe**ethernet** Protokoll ist sicherheitsgerichtet und TÜV zertifiziert bis SIL 3 gemäß IEC 61508.

Benötigte Ausstattung und Systemanforderungen:

HIMA Steuerung	HIMax mit CPU-Modul
Aktivierung	Diese Funktion ist bei allen HIMax Systemen standardmäßig freigeschaltet.

safeethernet (Eigenschaften):

Prozessormodule	safe**ethernet** wird auf dem CPU-Modul ausgeführt. Maximal 4 CPU-Module pro HIMax Steuerung.
Verbindungen	pro HIMax: maximal 255 safe**ethernet** Verbindungen pro HIMatrix: maximal 64 safe**ethernet** Verbindungen
Transportweg	Ethernet-Schnittstellen der CPU- und COM-Module Verwendete Ethernet-Schnittstellen simultan auch für andere Protokolle nutzbar.
Redundante Transportwege	2 Kanal Betrieb Redundante safe**ethernet** Transportwege zwischen HIMax und HIMax sind im safe**ethernet**-Editor einstellbar.
Prozessdatenmenge	Über jede safe**ethernet** Verbindung können je Richtung maximal 1100 Bytes übertragen werden (HIMatrix max. 900 Bytes). i Die maximale Prozessdatenmenge der Steuerung von 512 kByte darf nicht überschritten werden. In diesem Fall wird die Parametrierung der Steuerung beim Laden abgelehnt.
Projektübergreifende Kommunikation	safe**ethernet**-Verbindungen zu einer Ressource in einem anderen Projekt kann in SILworX konfiguriert werden, siehe Kapitel 4.8.

4.1 Was ist safeethernet

Im Bereich der Prozess- und Automatisierungstechnik sind Anforderungen wie Determinismus, Zuverlässigkeit, Austauschbarkeit, Erweiterbarkeit und vor allem Sicherheit zentrale Themen.

safe**ethernet** ist ein Übertragungsprotokoll zur Übertragung von sicherheitsgerichteten Daten bis SIL 3 auf Basis der Ethernet-Technologie.

safe**ethernet** beinhaltet Mechanismen, die folgende Fehler erkennen und darauf sicherheitsgerichtet reagieren:

- Verfälschung von übertragenen Daten (doppelte, verlorene, geänderte Bits)
- Falsche Adressierung von Nachrichten (Sender, Empfänger)
- Falsche Reihenfolge von Daten (Wiederholung, Verlust, Tausch)
- Falsches Zeitverhalten (Verzögerung, Echo)

safe**ethernet** basiert auf dem Standard IEEE 802.3.

Die Übertragung der sicherheitsrelevanten Daten verwendet den Standard-Ethernet-Protokollrahmen.

safe**ethernet** verwendet „unsichere Datenübertragungskanäle“ (Ethernet) nach dem Black Channel Prinzip und überwacht sie bei Sender und Empfänger durch sicherheitsgerichtete Protokollmechanismen. Dadurch sind Ethernet-Netzwerkkomponenten wie Hubs, Switches, Router innerhalb eines sicherheitsgerichteten Netzwerkes verwendbar.

Der markante Unterschied zu Standard-Ethernet besteht in der Sicherheit und der Echtzeitfähigkeit von safe**ethernet**. Ein spezieller Protokollmechanismus garantiert ein deterministisches Verhalten auch bei Ausfall oder Eintritt von Kommunikationsteilnehmern. Das System bindet neue Komponenten in das laufende System dann automatisch ein. Alle Komponenten eines Netzwerkes sind während des laufenden Betriebs austauschbar. Mit dem Einsatz von Switches lassen sich Übertragungszeiten klar definieren. Somit wird Ethernet echtzeitfähig.

Auch die mögliche Übertragungsgeschwindigkeit von bis zu 1 Gbit/s für sicherheitsgerichtete Daten liegt über den sonst gebräuchlichen Standards. Als Übertragungsmedien können Sie z. B. Kupferleitungen und LWL verwendet werden.

Verbindungen zum firmeninternen Intranet als auch Verbindungen zum Internet sind mit safe**ethernet** möglich. Damit ist nur noch ein Netzwerk für sichere und nicht sichere Datenübertragung nötig.

---

i Das Netzwerk darf von anderen Teilnehmern mitbenutzt werden, wenn genügend Übertragungskapazität zur Verfügung steht.

---

safe**ethernet** ermöglicht flexible Systemstrukturen für die dezentrale Automatisierung mit definierten Reaktionszeiten. Je nach Anforderung können Sie die Intelligenz wahlweise zentral oder dezentral auf die Teilnehmer innerhalb des Netzwerkes verteilen.

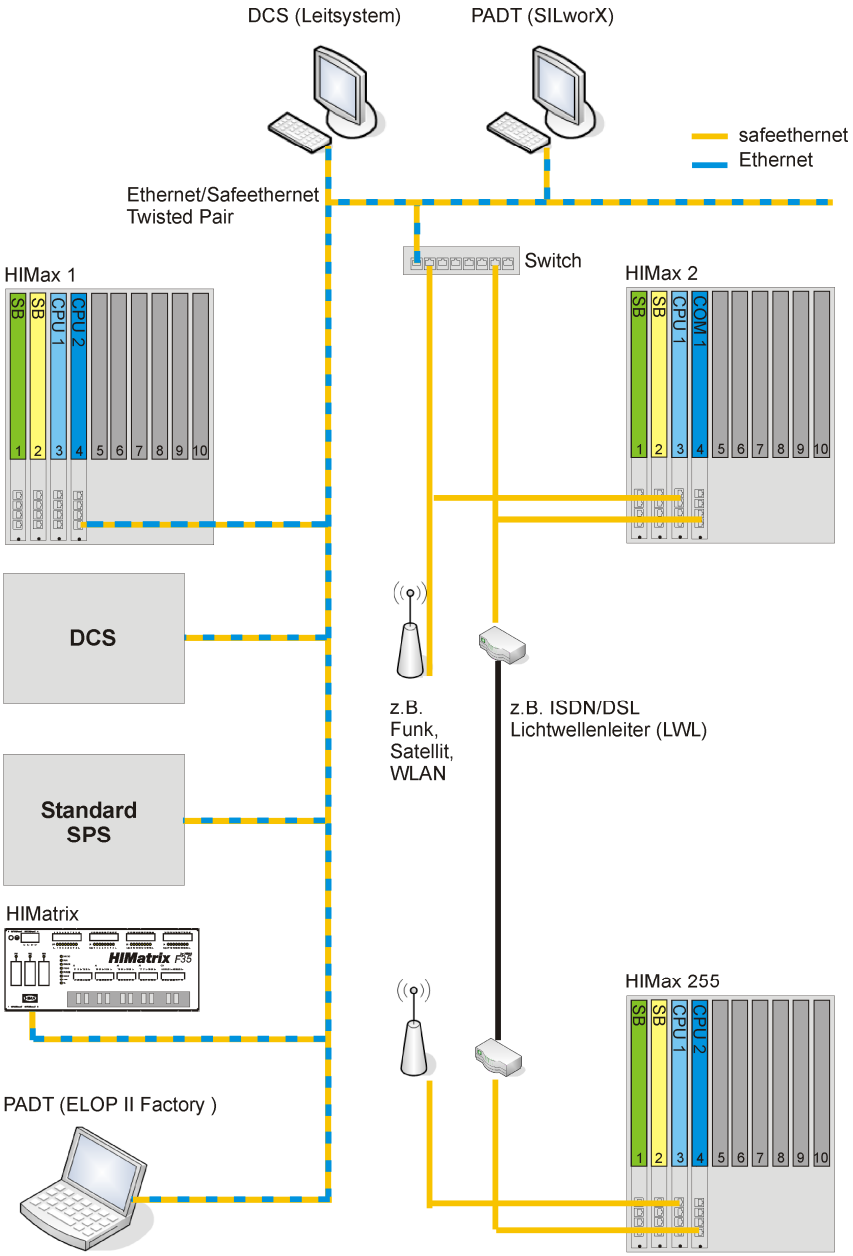


Bild 2: Systemstrukturen

i

Unbeabsichtigter Übergang in den sicheren Zustand möglich!

Bei der Zusammenschaltung ist zu beachten, dass keine Netzschleifen entstehen. Datenpakete dürfen nur auf einem Weg zu einer Steuerung gelangen.

4.2 Konfiguration einer redundanten safeethernet Verbindung

In diesem Beispiel konfigurieren Sie eine redundante HIMax/HIMax safe**ethernet**-Verbindung.

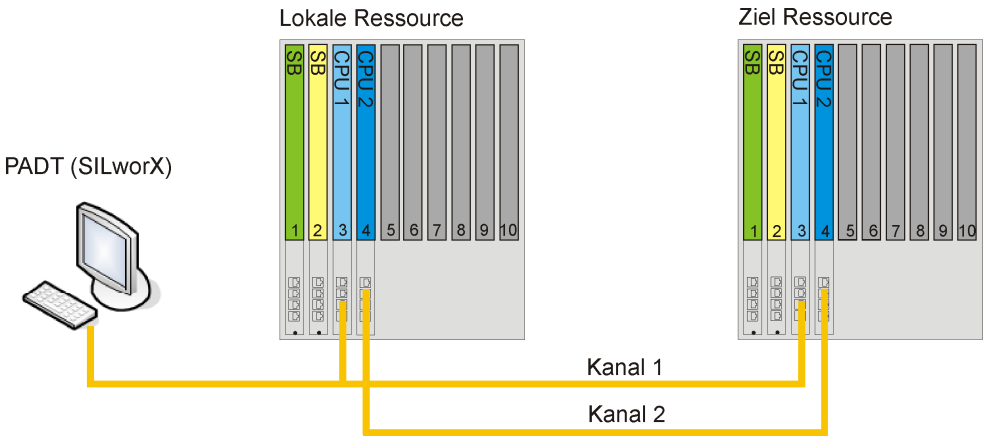


Bild 3: Aufbau zur Konfiguration einer redundante Verbindung

i HIMA empfiehlt, bei einer redundanten safe**ethernet** Verbindung die beiden Transportwege (Kanal 1 und Kanal 2) über separate Kommunikationsmodule zu führen. Dabei müssen Bandbreite und Verzögerung auf den jeweiligen Transportwegen annähernd identisch sein.

safe**ethernet**-Verbindung erstellen

Erstellen Sie im safe**ethernet**-Editor eine safe**ethernet**-Verbindung zwischen der lokalen Ressource und der Ziel-Ressource.

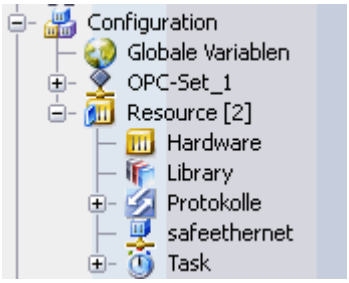


Bild 4: Strukturbaum der Ressource

So öffnen Sie den safeethernet Editor der lokalen Ressource:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf safe**ethernet** und im Kontextmenü **Edit** wählen.

   ☑ In der Objektauswahl befinden sich die Ziel-Ressourcen.

So erstellen Sie die safeethernet Verbindung zur Ziel-Ressource:

1. In der Objektauswahl auf **Ziel-Ressource** klicken und per Drag&Drop auf eine freie Stelle im Arbeitsbereich des safe**ethernet** Editors ziehen.

i Der umgekehrte Kommunikationspfad wird im safe**ethernet** Editor der Ziel-Ressource automatisch hinzugefügt.

So konfigurieren Sie die safeethernet-Verbindung:

1. **Ethernet-Schnittstellen Kanal 1** der Lokal- und Ziel-Ressource auswählen.
2. **Ethernet-Schnittstellen Kanal 2** der Lokal- und Ziel-Ressource auswählen.
3. **Netzwerk-Profil** (z. B. Fast&Noisy) der safe**ethernet** Verbindung auswählen.
4. **Receive Timeout** und **Response Time** berechnen und eintragen (siehe Kapitel 4.6).

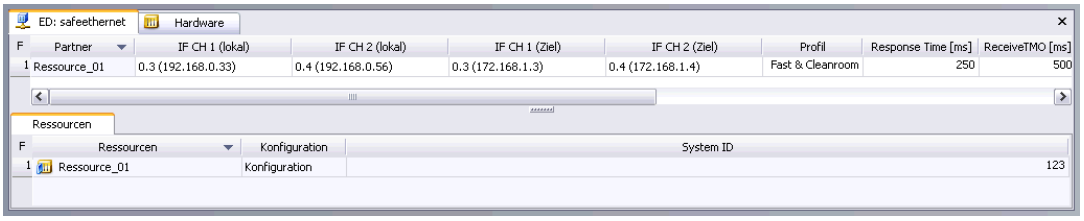


Bild 5: Werte der Parameter einer redundanten safe**ethernet**-Verbindung

4.2.1 Prozessvariablen verbinden

So öffnen Sie die Detailansicht einer safeethernet Verbindung:

Voraussetzung: safe**ethernet**-Editor der lokalen Ressource ist geöffnet.

1. Kontextmenü der **Ziel Ressource** durch Rechtsklick öffnen.
2. **Detailansicht** wählen.
3. Register **Ressource (Ziel)<->Ressource (lokal)** wählen.

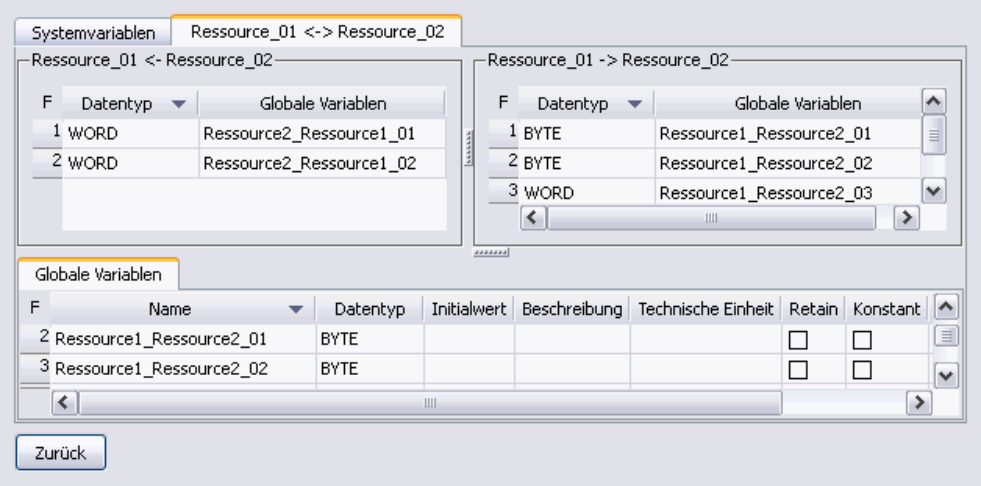


Bild 6: Detailansicht safe**ethernet** Editor

i Es können nur Globale Variablen aus dem Kontext der Konfiguration verwendet werden, nicht aus dem Kontext der Ressource!

So werden die safeethernet-Sendevariablen hinzugefügt:

Sendevariablen werden von der lokalen Ressource zur Ziel Ressource gesendet

1. Bereich **Ressource (Ziel)->Ressource (lokal)** wählen.
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in die Spalte **Ressource (Ziel)->Ressource (lokal)** ziehen.
3. Diesen Schritt für weitere safe**ethernet** Sendevariablen wiederholen.

So werden die safeethernet-Empfangsvariablen hinzugefügt:

Empfangsvariablen werden von der lokalen Ressource empfangen.

1. Bereich **Ressource (Ziel)<-Ressource (lokal)** wählen.
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in die Spalte **Ressource (Ziel)<-Ressource (lokal)** ziehen.
3. Diesen Schritt für weitere safe**ethernet** Empfangsvariablen wiederholen.

safeethernet Verbindung verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, safeethernet** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls Fehler korrigieren.

i Die Konfiguration der safe**ethernet**-Verbindung muss mit dem Anwenderprogramm der Lokalen- und Ziel-Ressource neu compiliert und in die Steuerungen übertragen werden, bevor sie für die Kommunikation der HIMax wirksam werden.

4.2.2 Prüfung der safeethernet Kommunikation

Setzen Sie im Control Panel die Anzeigen *Fehlerhafte Nachrichten* und *Wiederholungen* auf Null zurück.

1. Betreiben Sie das HIMax System unter voller Last:
   - Alle Kommunikationsprotokolle sind in Betrieb (safe**ethernet** und Standardprotokolle).
   - Prozessormodule ziehen und wieder stecken wie in Kapitel 4.6.1 beschrieben.
   - Anwenderprogramm per Reload laden.

i Zur Prüfung der korrekten Einrichtung einer redundanten safe**ethernet** Verbindung sollten Sie eine redundante Verbindung trennen und wieder hinzufügen und danach die andere. Dabei dürfen keine Fehler in der safe**ethernet** Kommunikation auftreten.

2. Überprüfen Sie in den Control Panels der beiden Steuerungen die Anzeigen *Fehlerhafte Nachrichten* und *Wiederholungen*.
   Sind die Zähler von *Fehlerhafte Nachrichten* und *Wiederholungen*
   = 0, sind die safe**ethernet** Einstellungen OK.
   ≥ **0**, müssen die safe**ethernet** Einstellungen nochmals geprüft werden.
   - Berechnen Sie mit der maximalen Zykluszeit die *Receive Timeout* neu siehe Kapitel 4.6.1 und 4.6.2.
   - Variieren Sie die *Response Time* wie in Kapitel 4.6.3 beschrieben.

i Weitere Ursachen für *Fehlerhafte Nachrichten* und *Wiederholungen!*

Prüfen Sie den korrekten Netzwerkaufbau (z. B. Leitungen, Switches, PCs).
Wird das Ethernet-Netzwerk nicht exklusiv für safe**ethernet** verwendet, ist zudem die Netwerkauslastung (Wahrscheinlichkeit von Datenkollisionen) zu prüfen.

4.3 safeethernet-Editor

Im safe**ethernet**-Editor erstellen und konfigurieren Sie die safe**ethernet**-Verbindungen zu den Kommunikationspartnern (Ressourcen).

So öffnen Sie den safeethernet Editor der lokalen Ressource:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf safe**ethernet** und im Kontextmenü **Edit** wählen.
   ☑ Der safe**ethernet**-Editor enthält den Arbeitsbereich und die Objektauswahl.

Im safe**ethernet**-Editor erstellen und konfigurieren Sie die safe**ethernet** Verbindungen zu den Kommunikationspartnern (Ressourcen). Dazu ziehen Sie die Ressourcen aus der Objektauswahl in den Arbeitsbereich.

Zur Konfiguration der safe**ethernet**-Verbindung müssen Sie die folgenden safe**ethernet**-Protokoll-Parameter einstellen:

Parameter	Beschreibung
Partner	Ressource-Name des Linkpartners
IF CH...	Verfügbare Ethernet-Schnittstellen auf der Ressource (lokal) und Ressource (Ziel), siehe auch Kapitel 3.6.
Profil	Kombination zueinander passender safe**ethernet** Parameter, siehe auch Kapitel 4.7.5.
Sync/Async	Datenaustausch synchron oder asynchron zum CPU-Zyklus einstellbar, siehe auch Kapitel 4.6.2.
Response Time [ms]	Zeit bis zur Empfangsbestätigung einer Nachricht beim Absender, siehe auch Kapitel 4.6.3.
Receive Timeout [ms]	Überwachungszeit auf PES1, innerhalb der eine korrekte Antwort von PES2 empfangen werden muss, siehe auch Kapitel 4.6.2.
Resend Timeout [ms]	Überwachungszeit auf PES1, innerhalb welcher PES2 den Empfang eines Datenpaketes bestätigt haben muss, ansonsten wird das Datenpaket wiederholt, siehe auch Kapitel 4.6.5.
Acknowledge Timeout [ms]	Zeit, nach der ein empfangenes Datenpaket von der CPU spätestens bestätigt werden muss, siehe auch Kapitel 4.6.6.
Prod.-Rate	Produktionsrate: Kleinstes Zeitintervall zwischen zwei Datenpaketen, siehe auch Kapitel 4.6.7.
Speicher (Queue-Tiefe)	Anzahl der Datenpakete, die ohne Empfangsbestätigung versendet werden können, siehe auch Kapitel 4.6.8.

Freeze-Daten bei Verbindungsverlust [ms]	Verhalten der Input Variablen dieser safe**ethernet** Verbindung bei Verbindungsunterbrechung. Verwende Initialdaten: Für die Input Variablen werden die Initialdaten verwendet. Unbegrenzt: Die Input Variablen werden auf dem momentanen Wert eingefroren und bis zur erneuten Verbindungsaufnahme verwendet. Begrenzt: Eingabe: Doppelklick auf Dropdown-Feld und Zeit eingeben. Die Input Variablen werden auf dem momentanen Wert eingefroren und bis nach dem parametrierten Timeout verwendet. Danach werden die Initialdaten verwendet. Der Start des Timeout kann um bis zu einem CPU-Zyklus verzögert werden. **⚠ VORSICHT** **Für sicherheitsgerichtete Funktionen, die über safeethernet realisiert werden, darf nur die Einstellung *Verwende Initialdaten* benutzt werden.**
Fragmente pro Zyklus	Feste Einstellung: Ein View wird pro Zyklus der Steuerung zum Kommunikationspartner übertragen. Ein View einer HIMax ist ein Fragment ≤ 1100 Byte Ein View einer HIMatrix ist ein Fragment ≤ 900 Byte
Anzahl ignorierte Warnungen	Noch keine Funktion!
Zeitraum Warnungen [ms]	Noch keine Funktion!
SER aktivieren	Standardwert: aktiviert

Tabelle 25: Parameter safe**ethernet**-Protokoll

Objektauswahl

Die Objektauswahl stellt alle Ressourcen innerhalb dieses Projektes zur Verfügung, mit denen diese Ressource über safe**ethernet** verbunden werden kann.

i Für safe**ethernet** Verbindungen zu Ressourcen außerhalb eines Projektes oder zu einer (in ELOP II Factory projektierten) HIMatrix Steuerung steht eine Exportfunktion zur Verfügung (siehe Kapitel 4.8).

4.4 Detailansicht des safeethernet-Editors

Die **Detailansicht** hat immer den Bezug auf die lokale Ressource, für die Sie den safe**ethernet**-Editor gestartet haben.

So öffnen Sie die Detailansicht einer safeethernet Verbindung:

1. Mit Rechtsklick auf safe**ethernet** Verbindung Kontextmenü öffnen.
2. Auf **Detailansicht** klicken.
   ☑ Die **Detailansicht** beinhaltet das Register **Systemvariablen, View-Definitionen** und **Ressource (lokal)<->Ressource (Ziel)**.

4.4.1 Register: Systemvariablen

Sie können die safe**ethernet** Verbindung im Anwenderprogramm mit Hilfe von Systemvariablen steuern und deren Status auswerten.

<table>
<tr><th>Systemvariable</th><th>Beschreibung</th></tr>
<tr><td>Ack-Frame-Nr.</td><td>Empfangszähler (Umlaufend).</td></tr>
<tr><td>Anzahl defekter Nachrichten</td><td rowspan="2">Anzahl aller defekter Nachrichten pro Kanal (falscher CRC, falscher Header, sonstige Fehler)</td></tr>
<tr><td>Anzahl defekter Nachrichten des Red. Kanal</td></tr>
<tr><td>Anzahl Verbindungserfolge</td><td>Anzahl der Verbindungserfolge seit Reset der Statistik.</td></tr>
<tr><td>Anzahl verlorener Nachrichten</td><td rowspan="2">Anzahl der auf einem der beiden Transportwege ausgefallenen Nachrichten seit Reset der Statistik.
Der Zähler wird nur bis zum Komplettausfall eines Kanals geführt.</td></tr>
<tr><td>Anzahl verlorener Nachrichten des Red.-Kanal</td></tr>
<tr><td>Early Queue Usage</td><td>Anzahl der Nachrichten die in Early Queue gelegt wurden seit Reset der Statistik, siehe auch Kapitel 4.6.8.</td></tr>
<tr><td>Fehlerhafte Nachrichten</td><td>Anzahl verworfener Nachrichten seit Reset der Statistik.</td></tr>
<tr><td>Frame-Nr</td><td>Sendungszähler (Umlaufend).</td></tr>
<tr><td>Kanalzustand</td><td>Aktueller Kanalzustand von Kanal 1.
Der Kanalzustand ist der aktuelle Zustand des Kanal 1 zum Zeitpunkt (Seq-No X-1) beim Empfang einer Nachricht mit Seq-No X.
<table>
<tr><th>Status</th><th>Beschreibung</th></tr>
<tr><td>0</td><td>Keine Info zum Zustand von Kanal 1.</td></tr>
<tr><td>1</td><td>Kanal 1 OK.</td></tr>
<tr><td>2</td><td>Letzte Nachricht war Fehlerhaft, aktuelle ist OK.</td></tr>
<tr><td>3</td><td>Fehler auf Kanal 1.</td></tr>
</table></td></tr>
<tr><td>Layoutversion</td><td>Signatur des in der Kommunikation verwendeten Datenlayouts.</td></tr>
<tr><td>Letzte Kanal Latenz</td><td rowspan="8">Die Kanal Latenz gibt die Verzögerung zwischen beiden redundanten Transportpfaden zum Empfangszeitpunkt von Nachrichten mit identischer SeqNo an.
Hierfür wird eine Statistik mit durchschnittlicher, minimaler, maximaler und letzter Latenz geführt.
Ist der Min-Wert > dem Max-Wert, so sind die Statistikwerte ungültig.
Letzte Kanal Latenz und Mittlere Kanal Latenz sind dann 0.</td></tr>
<tr><td>Letzte Latenz des Red.-Kanal</td></tr>
<tr><td>Max. Kanal Latenz</td></tr>
<tr><td>Max. Kanal Latenz des Red. Kanal</td></tr>
<tr><td>Min. Kanal Latenz</td></tr>
<tr><td>Min. Kanal Latenz des Red. Kanal</td></tr>
<tr><td>Mittlere Kanal Latenz</td></tr>
</table>

<table>
<tr><td>Mittlere Kanal Latenz des Red. Kanal</td><td></td></tr>
<tr><td>Monotonie</td><td>Nutzdatensendungszähler (Umlaufend).</td></tr>
<tr><td>Neue Layoutversion</td><td>Signatur des neuen Datenlayouts.</td></tr>
<tr><td>Qualität Kanal 1</td><td>Zustand des Haupt-Transportweges.
<table>
<tr><th>Bit Nr.</th><th>Bit = 0</th><th>Bit = 1</th></tr>
<tr><td>0</td><td>Transportweg nicht freigegeben</td><td>Transportweg freigegeben</td></tr>
<tr><td>1</td><td>Transportweg nicht genutzt</td><td>Transportweg aktiv genutzt</td></tr>
<tr><td>2</td><td>Transportweg nicht verbunden</td><td>Transportweg verbunden</td></tr>
<tr><td>3</td><td>-</td><td>Transportweg liefert Nachricht zuerst</td></tr>
<tr><td>4 - 7</td><td>Reserviert</td><td>Reserviert</td></tr>
</table></td></tr>
<tr><td>Qualität Kanal 2</td><td>Zustand des redundanten Transportweges, siehe Zustand Kanal 1 (Haupt-Transportweg).</td></tr>
<tr><td>Receive Timeout</td><td>Zeit in Millisekunden (ms) auf PES1, innerhalb der eine gültige Antwort von PES2 empfangen werden muss, siehe auch Kapitel 4.6.2.</td></tr>
<tr><td>Response Time</td><td>Zeit in Millisekunden (ms) bis zur Empfangsbestätigung einer Nachricht beim Absender, siehe auch Kapitel 4.6.3</td></tr>
<tr><td>safeethernet-Statistik Zurücksetzen</td><td>Statistikwerte für die Kommunikationsverbindung im Anwenderprogramm Rücksetzen. Dazu Parameter Statistik zurücksetzen von FALSE auf TRUE stellen.</td></tr>
<tr><td>Transport-Steuerung Kanal1</td><td>Transportsteuerung von Kanal1
<table>
<tr><th>Bit 0</th><th>Funktion</th></tr>
<tr><td>FALSE</td><td>Transportweg freigegeben</td></tr>
<tr><td>TRUE</td><td>Transportweg gesperrt</td></tr>
</table>
<table>
<tr><th>Bit 1</th><th>Funktion</th></tr>
<tr><td>FALSE</td><td>Transportweg für Tests freigegeben</td></tr>
<tr><td>TRUE</td><td>Transportweg gesperrt</td></tr>
</table>
Bit 2 ... 7 reserviert.</td></tr>
<tr><td>Transport-Steuerung Kanal2</td><td>Transportsteuerung von Kanal 2, siehe Transportsteuerung Kanal 1.</td></tr>
</table>

<table>
<tr><td>Verbindungssteuerung</td><td>Mit dieser Systemvariablen kann die safeethernet-Verbindung vom Anwenderprogramm gesteuert werden.
<table>
<tr><th>Befehl</th><th>Beschreibung</th></tr>
<tr><td>Autoconnect (0x0000)</td><td>Standardwert:
Nach Verlust der safeethernet Kommunikation versucht die Steuerung im nächsten CPU-Zyklus, die Verbindung wieder aufzunehmen.</td></tr>
<tr><td>Toggle Mode 0 (0x0100)
Toggle Mode 1 (0x0101)</td><td>Nach dem Kommunikationsverlust kann durch einen programmgesteuerten Wechsel des Toggle Modus die Verbindung erneut aufgebaut werden.
▪ TOGGLE MODE_0 (0x100) gesetzt:
Auf TOGGLE MODE 1 (0x101) setzen um die Verbindung wieder aufzunehmen.
▪ TOGGLE MODE 1 (0x101) gesetzt:
Auf TOGGLE_MODE_0 (0x100) setzen um die Verbindung wieder aufzunehmen.</td></tr>
<tr><td>Disabled (0x8000)</td><td>safeethernet Kommunikation abgeschaltet.</td></tr>
</table></td></tr>
<tr><td>Verbindungszustand</td><td>Der Verbindungszustand wertet den Status der Kommunikation zwischen zwei Steuerungen im Anwenderprogramm aus.
<table>
<tr><th>Status/Wert</th><th>Beschreibung</th></tr>
<tr><td>Closed (0)</td><td>Verbindung ist geschlossen und es wird auch nicht versucht sie zu öffnen.</td></tr>
<tr><td>Try_open (1)</td><td>Verbindung wird versucht zu öffnen, sie ist jedoch noch nicht geöffnet. Dieser Zustand gilt gleichermaßen für die aktive und auch für die passive Seite.</td></tr>
<tr><td>Connected (2)</td><td>Die Verbindung ist hergestellt und in Betrieb (aktive Zeitüberwachung und Datenaustausch)</td></tr>
</table></td></tr>
<tr><td>Wiederholungen</td><td>Anzahl der Wiederholungen seit Reset der Statistik.</td></tr>
<tr><td>Zeitstempel des letzten Fehlers des Red.-Kanal [ms]</td><td>Millisekunden Anteil des Zeitstempels (aktuelle Systemzeit).</td></tr>
<tr><td>Zeitstempel des letzten Fehlers des Red.-Kanals [s]</td><td>Sekunden Anteil des Zeitstempels (aktuelle Systemzeit).</td></tr>
<tr><td>Zeitstempel des letzten Fehlers [ms]</td><td>Millisekunden Anteil des Zeitstempels (aktuelle Systemzeit).</td></tr>
<tr><td>Zeitstempel des letzten Fehlers [s]</td><td>Sekunden Anteil des Zeitstempels (aktuelle Systemzeit).</td></tr>
</table>

<table>
<tr><td>Zustand des Red.-Kanal</td><td>Aktueller Kanalzustand von Kanal 2.
Der Kanalzustand ist der aktuelle Zustand des Kanal 2 zum Zeitpunkt (Seq-No X-1) beim Empfang einer Nachricht mit Seq-No X.
<table>
<tr><th>Status</th><th>Beschreibung</th></tr>
<tr><td>0</td><td>Keine Info zum Zustand von Kanal 2</td></tr>
<tr><td>1</td><td>Kanal 2 OK.</td></tr>
<tr><td>2</td><td>Letzte Nachricht war Fehlerhaft, aktuelle ist OK.</td></tr>
<tr><td>3</td><td>Fehler auf Kanal 2.</td></tr>
</table>
</td></tr>
</table>

Tabelle 26: Register Systemvariablen im safe**ethernet**-Editor

4.5 Mögliche safeethernet Verbindungen

Sie können eine safe**ethernet**-Verbindung zwischen zwei HIMax Steuerungen, mono oder redundant aufbauen.

Die verfügbaren Ethernet-Schnittstellen für eine safe**ethernet** Verbindung werden immer in Bezug auf die Ressource (lokal) angezeigt, für die der safe**ethernet** Editor gestartet wurde.

Alle verfügbaren Ethernet-Schnittstellen einer Steuerung werden in den Dropdown-Menüs der Parameter **IF CH…** angezeigt.

Element	Beschreibung
IF CH1 (lokal)	Ethernet-Schnittstelle (Kanal 1) der Ressource
IF CH2 (lokal)	Ethernet-Schnittstelle (Kanal 2) der Ressource
IF CH1 (Ziel)	Ethernet-Schnittstelle (Kanal 1) des Linkpartners
IF CH2 (Ziel)	Ethernet-Schnittstelle (Kanal 2) des Linkpartners

Tabelle 27: Verfügbare Ethernet-Schnittstellen

4.5.1 Mono safeethernet Verbindung (Kanal 1)

Für eine Mono-Verbindung müssen Sie die Ethernet-Schnittstellen *IF CH1 (lokal)* und *IF CH1 (Ziel)* in der lokalen Ressource setzen.

4.5.2 Redundante safeethernet Verbindung (Kanal 1 und Kanal 2)

Redundante safe**ethernet** Transportwege zwischen zwei HIMax/HIMax Steuerungen sind möglich.

i Bandbreite und Verzögerung auf den beiden Transportwegen müssen annähernd identisch sein.

Für eine redundante Verbindung sind die folgenden Ethernet-Schnittstellen benutzbar:

- Die Ethernet-Schnittstellen *IF CH1 (lokal)* und *IF CH1 (Ziel)* für Kanal 1.
- Die Ethernet-Schnittstellen *IF CH2 (lokal)* und *IF CH2 (Ziel)* für Kanal 2.
- Für eine redundante Verbindung über Kanal1 und Kanal2 über nur eine Ethernet-Schnittstelle wählen Sie im safe**ethernet**-Editor dieselbe Ethernet-Schnittstelle in *IF CH1 (lokal)* Kanal1 und *IF CH2 (lokal)* Kanal2 aus.

i Der umgekehrte Kommunikationspfad wird im safe**ethernet** Editor der Ziel-Ressource automatisch hinzugefügt.

4.5.3 Erlaubte Kombinationen

In der folgenden Tabelle sind die möglichen Kombinationen für eine redundante safe**ethernet** Verbindung dargestellt.

Kanal 1 IF Ch1 (lokal) / IF Ch1 (Ziel)	Kanal 2 IF Ch2 (lokal) / IF Ch2 (Ziel)
CPU1/CPU1	CPU2/CPU2
CPU1/CPU1	CPU1/CPU1
COM1/COM1	COM2/COM2
CPU1/COM1	CPU2/COM2
CPU1/COM2	CPU2/COM1
CPU1/CPU1	COM1/COM1

Tabelle 28: Kombinationen für safe**ethernet**-Verbindungen

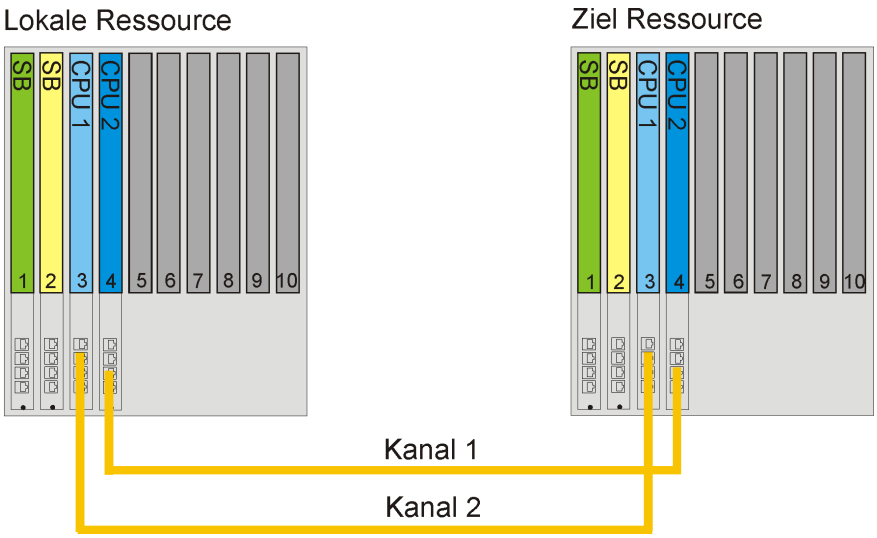


Bild 7: Redundante Verbindung zwischen zwei HIMax Steuerungen

Folgende redundant konfigurierte Verbindung (über eine Leitung) ist sinnvoll um einen konstanten Datendurchsatz trotz verloren gegangener Datenpakete (z. B. durch EMV-Störungen) zu erlangen.

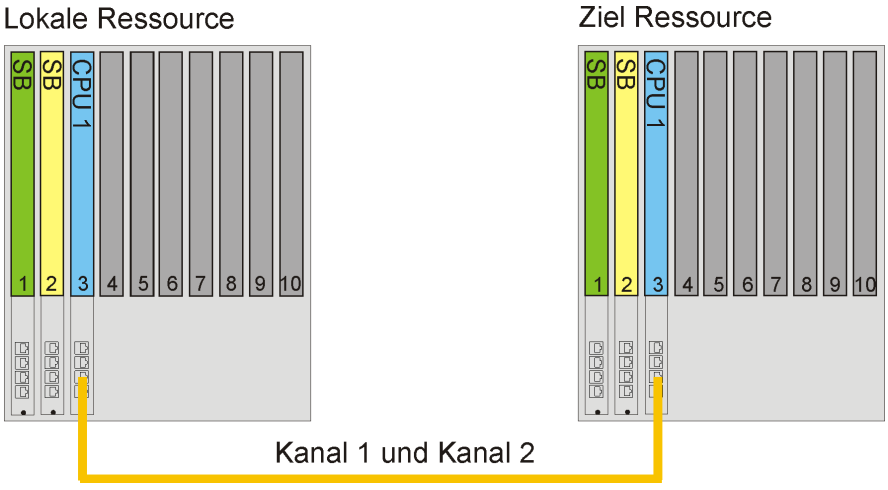


Bild 8: Redundante Verbindung zweier HIMax Steuerungen über eine Leitung

4.6 safeethernet Parameter

Die sicherheitsgerichtete Kommunikation richten Sie im safe**ethernet**-Editor ein. Dazu müssen Sie die in diesem Kapitel beschriebenen Parameter parametrieren.

Für die Berechnung der safeethernet Parameter *Receive Timeout* und *Response Time* gilt folgende Bedingung:
Die Kommunikations-Zeitscheibe muss ausreichend groß sein, um in einem CPU-Zyklus alle safe**ethernet** Verbindungen abzuarbeiten, siehe Kapitel 12.2.

4.6.1 Maximale Zykluszeit (minimale Watchdog-Zeit) der HIMax Steuerung

Zur Bestimmung der maximalen Zykluszeit (minimale Watchdog-Zeit) für eine HIMax Steuerung empfiehlt HIMA die folgende Vorgehensweise an einem vollständigen System, bei dem alle Prozessormodule gesteckt sind.

1. Watchdog-Zeit zum Test hoch einstellen.
2. System unter voller Last betreiben. Dabei müssen alle Kommunikationsverbindungen in Betrieb sein, sowohl über safeethernet als auch über Standardprotokolle. Die Zykluszeit im Control Panel öfter ablesen, und die Schwankungen der Zykluszeit notieren.
3. Nacheinander jedes Prozessormodul entfernen und wieder in den Basisträger einfügen. Jeweils vor dem Entfernen eines Prozessormoduls warten, bis sich das gerade eingefügte Prozessormodul aufsynchronisiert hat.

i Beim Hinzufügen eines Prozessormoduls synchronisiert sich dieses automatisch mit der Konfiguration der vorhandenen Prozessormodule. Die für die Synchronisation benötigte Zeit verlängert den Zyklus der Steuerung auf die maximale Zykluszeit.

Die für die Synchronsation benötigte Zeit wächst mit der Anzahl der bereits synchronisierten Prozessormodule.

Beschreibung zum Ein- und Ausbau eines Prozessormoduls siehe Handbuch X-CPU 01, HI 801 008.

4. In der Diagnosehistorie die Synchronisationszeit von n auf n+1 Prozessormodule bei jedem Synchronisationsvorgang ablesen.
5. Wiederholen Sie diese Schritte für den Kommunikationspartner (zweite HIMax Steuerung). Die größte dieser Synchronisationszeiten wird zur Bestimmung der Watchdog-Zeit benutzt.

i Notieren Sie sich die Synchronisationszeiten beider HIMax-Steuerungen!

6. Minimale Watchdog-Zeit berechnen aus:
größte Synchronisationszeit + 12 ms Reserve + Reserve für die notierten Schwankungen.

Damit ist eine geeignete Vorgabe der maximalen Zykluszeit (minimale Watchdog-Zeit) für die nachfolgenden Berechnungen ermittelt.

TIPP Führen Sie die Berechnung im Schritt 6 für beide HIMax Steuerungen, mit den jeweiligen notierten Synchronisationszeiten aus.

Die auf diese Weise ermittelten maximale Zykluszeiten (minimale Watchdog-Zeiten) können für die Watchdog-Zeit der jeweiligen Ressource verwendet werden, siehe Sicherheitshandbuch HI 801 002.

4.6.2 Receive Timeout

ReceiveTMO ist die Überwachungszeit in Millisekunden (ms), innerhalb der eine korrekte Antwort des Kommunikationspartners empfangen werden muss.

Trifft innerhalb der *ReceiveTMO* keine korrekte Antwort des Kommunikationspartners ein, wird die sicherheitsgerichtete Kommunikation geschlossen. Die Input Variablen dieser safe**ethernet** Verbindung verhalten sich gemäß dem eingestellten Parameter *Freeze-Daten bei Verbindungsverlust [ms]*.

Für sicherheitsgerichtete Funktionen, die über safe**ethernet** realisiert werden, darf nur die Einstellung **Verwende Initialdaten** benutzt werden..

Da die *ReceiveTMO* sicherheitsrelevant und Bestandteil der Worst Case Reaction Time T_R (maximale Reaktionszeit, siehe Kapitel 4.7.1ff) ist, muss die *ReceiveTMO* wie folgt berechnet und im safe**ethernet** Editor eingetragen werden.

ReceiveTMO ≥ 4*Delay + 5*max. Zykluszeit

Bedingung: Die Kommunikations-Zeitscheibe muss ausreichend groß sein, um in einem CPU-Zyklus alle safe**ethernet** Verbindungen abzuarbeiten.

Delay:	Verzögerung auf der Übertragungsstrecke, z.B. durch (Switch, Satellit usw.)
max. Zykluszeit:	maximale Zykluszeit der beiden Steuerungen

i Eine erwünschte Fehlertoleranz der Kommunikation kann über eine Erhöhung der *ReceiveTMO* erreicht werden, sofern dies für den Anwendungsprozess zeitlich zulässig ist, siehe Kapitel 4.7.

4.6.3 ResponseTime

Die *ResponseTime* ist die Zeit in Millisekunden (ms), die verstreicht, bis der Absender einer Nachricht die Empfangsbestätigung des Empfängers erhält.

Für die Parametrierung unter Verwendung eines safe**ethernet** Profils muss eine durch die physikalischen Gegebenheiten der Übertragungsstrecke erwartete *ResponseTime* vorgegeben werden.

Die vorgegebene *ResponseTime* hat Einfluss auf die Konfiguration aller Parameter der safe**ethernet** Verbindung, die Sie wie folgt berechnen:

ResponseTime ≤ ReceiveTMO / n

n = 2, 3, 4, 5, 6, 7, 8.....

Das Verhältnis der *ReceiveTMO* und der *ResponseTime* beeinflusst die Fähigkeit zur Fehlertoleranz, z. B. bei Paketverlusten (Wiederholung von verloren gegangenen Datenpaketen) oder Verzögerungen auf dem Übertragungsweg.

In einem Netzwerk, in dem es zu Paketverlusten kommen kann, muss die folgende Bedingung erfüllt sein:

min. Response Time ≤ ReceiveTMO / 2 ≥ 2*Delay + 2,5*max. Zykluszeit

Ist diese Bedingung erfüllt, kann der Verlust wenigstens eines Datenpaketes abgefangen werden, ohne dass die safe**ethernet** Verbindung unterbrochen wird.

i Ist diese Bedingung nicht erfüllt, kann die Verfügbarkeit einer safe**ethernet** Verbindung nur in einem kollisions- und störungsfreien Netzwerk garantiert werden. Dies bedeutet jedoch kein Sicherheitsproblem für das Prozessormodul!

i Stellen Sie sicher, dass das Kommunikationssystem die parametrierte Response-Time einhält!

Falls Sie dies nicht immer garantieren können, steht ihnen zur Überwachung der Response-Time eine entsprechende Systemvariable der Verbindung zur Verfügung. Kommt es nicht nur in seltenen Einzelfällen zu einer Überschreitung der gemessenen Response-Time über die halbe ReceiveTMO, muss die parametrierte Response Time erhöht werden.

Die Receive Timeout ist der neu parametrierten Response Time anzupassen.

4.6.4 Sync/Async

Sync	Zur Zeit nicht unterstützt.
Async	Ist die Standardeinstellung. Bei der Einstellung *Async* empfängt die safe**ethernet** Protokolleinstanz in der Input-Phase der CPU und sendet gemäß ihren Senderegeln in der Output-Phase der CPU.

4.6.5 ResendTMO

ResendTMO kann nicht manuell eingegeben werden, sondern wird aus dem Profil und der Response-Time berechnet.

Überwachungszeit in Millisekunden (ms) auf PES1, innerhalb welcher PES2 den Empfang eines Datenpaketes bestätigt haben muss, ansonsten wird das Datenpaket wiederholt.

Regel: *ResendTMO* ≤ *Receive-Timeout*

Bei unterschiedlicher Konfiguration der *Resend-Timeout* bei den Kommunikationspartnern bestimmt der aktive Protokollpartner (kleinere SRS) den tatsächlichen Wert der *Resend-Timeout* der Protokollverbindung.

4.6.6 Acknowledge Timeout

AckTMO kann nicht manuell eingegeben werden, sondern wird aus dem Profil und der Response-Time berechnet.

AckTMO ist die Zeit, nach der ein empfangenes Datenpaket von der CPU spätestens bestätigt werden muss.

Für ein schnelles Netzwerk ist *AckTMO* null, d.h. der Empfang eines Datenpaketes wird sofort bestätigt. Für ein langsames Netzwerk (z. B. Telefonmodemstrecke) ist *AckTMO* größer null. In diesem Fall wird versucht, die Bestätigungsmeldung zusammen mit Prozessdaten zu übermitteln, um die Netzbelastung durch Vermeidung von Adressierungs- und Sicherungsblöcken zu reduzieren.

Regeln:

- *AckTMO* muss ≤ *Receive-Timeout*

- *AckTMO* muss ≤ *Resend-Timeout* sein, wenn *Production-Rate* > *Resend-Timeout* ist.

4.6.7 Production Rate

ProdRate kann nicht manuell eingegeben werden, sondern wird aus dem Profil und der Response-Time berechnet.

Kleinstes Zeitintervall in Millisekunden (ms) zwischen zwei Datenpaketen.

Das Ziel von *Prod.-Rate* ist, die Menge an Datenpaketen auf ein Maß zu begrenzen, welches einen (langsamen) Kommunikationskanal nicht überlastet. Dadurch wird eine gleichmäßige Auslastung des Übertragungsmediums erreicht und der Empfang veralteter Daten auf der Empfängerseite vermieden.

Regeln:

- *ProdRate* ≤ *Receive-Timeout*
- *ProdRate* ≤ *Resend-Timeout*, wenn *Acknowledge-Timeout* > *Resend-Timeout*

i Eine Production Rate von null bedeutet, dass mit jedem Zyklus des Anwenderprogramms Datenpakete übertragen werden kann.

4.6.8 Speicher

Speicher kann nicht manuell eingegeben werden, sondern wird aus dem Profil und der Response-Time berechnet.

Speicher (Queue-Tiefe) ist die Anzahl der Datenpakete, die ausgesendet werden können, ohne auf deren Empfangsbestätigung warten zu müssen.

Der Wert ist abhängig von der Übertragungskapazität des Netzwerkes und möglichen Verzögerungen durch Netzwerklaufzeiten.

Alle safe**ethernet** Verbindungen teilen sich den zur Verfügung stehenden Message-Speicher in der CPU.

4.7 Maximale Reaktionszeit für safeethernet

In den folgenden Beispielen gelten die Formeln für die Berechnung der maximalen Reaktionszeit im Fall einer Verbindung mit HIMatrix Steuerungen nur dann, wenn auf diesen die Sicherheitszeit = 2 * Watchdog-Zeit eingestellt ist. Für HIMax Steuerungen gelten diese Formeln immer.

i Die zulässige maximale Reaktionszeit ist abhängig vom Prozess und ist mit der abnehmenden Prüfstelle abzustimmen.

Begriffe:

ReceiveTMO:	Überwachungszeit im PES 1, in der eine gültige Antwort vom PES 2 empfangen werden muss. Nach Ablauf der Zeit wird die sicherheitsgerichtete Kommunikation andernfalls geschlossen.
Production Rate:	Mindestabstand zwischen zwei Datensendungen.
Watchdog-Zeit:	Maximal erlaubte Dauer eines RUN-Zyklus in einer Steuerung. Die Dauer des RUN-Zyklus hängt von Komplexität des Anwenderprogramms und der Anzahl der safe**ethernet** Verbindungen ab. Watchdog-Zeit (WDZ) ist in den Eigenschaften der Ressource einzutragen.
Worst Case Reaction Time:	Maximale Reaktionszeit für die Übertragung der Änderung des Signals eines physikalischen Einganges (In) eines PES 1 bis zur Änderung des physikalischen Ausgangs (Out) eines PES 2.
Delay:	Verzögerung einer Übertragungsstrecke z. B. bei Modem- oder Satellitenverbindung. Bei direkter Verbindung kann zunächst eine Verzögerung von 2 ms angenommen werden. Die tatsächliche Verzögerung der Übertragungsstrecke kann von dem zuständigen Netzwerkadministrator ausgemessen werden.

Für die folgenden Berechnungen der zulässigen maximalen Reaktionszeiten gelten folgende Bedingungen:

- Die Signale, die mit safe**ethernet** übertragenen werden, müssen in den jeweiligen Steuerungen innerhalb eines CPU-Zyklus verarbeitet werden.
- die Reaktionszeiten der Sensoren und Aktoren sind zusätzlich zu addieren.

Die Berechnungen gelten auch für Signale in umgekehrter Richtung.

4.7.1 Berechnung der maximalen Reaktionszeit zweier HIMax PES

Maximale Reaktionszeit T_R („Worst Case“) vom Wechsel eines Gebers des PES 1 (In) bis zur Reaktion des Ausgangs (Out) des PES 2 wie folgt berechnen:

Bild 9: safe**ethernet** Verbindung zweier HIMax Steuerungen

Reaktionszeit bei Verbindung zweier HIMax Steuerungen

$T_R = t_1 + t_2 + t_3$

T_R Worst Case Reaction Time

t_1 Sicherheitszeit des PES 1

t_2 *ReceiveTMO*

t_3 Sicherheitszeit des PES 2

4.7.2 Berechnung der max. Reaktionszeit in Verbindung mit einem HIMatrix-PES

Maximale Reaktionszeit T_R („Worst Case“) vom Wechsel eines Gebers (In) des HIMax PES 1 bis zur Reaktion des Ausgangs (Out) des HIMatrix PES 2 wie folgt berechnen:

Bild 10: safe**ethernet** Verbindung einer HIMax mit einer HIMatrix-Steuerung

Reaktionszeit bei Verbindung einer HIMax mit einer HIMatrix-Steuerung

$T_R = t_1 + t_2 + t_3$

T_R Worst Case Reaction Time

t_1 Sicherheitszeit des HIMax PES 1

t_2 *ReceiveTMO*

t_3 2 * Watchdog-Zeit des HIMatrix-PES 2

4.7.3 Berechnung der max. Reaktionszeit mit zwei HIMatrix-Steuerungen oder RIOs

Maximale Reaktionszeit T_R („Worst Case“) vom Wechsel eines Gebers (In) im ersten HIMatrix PES oder in RIO (z. B. F3 DIO 20/8 01) bis zur Reaktion des Ausgangs im zweiten HIMatrix PES oder in RIO (Out) wie folgt berechnen:

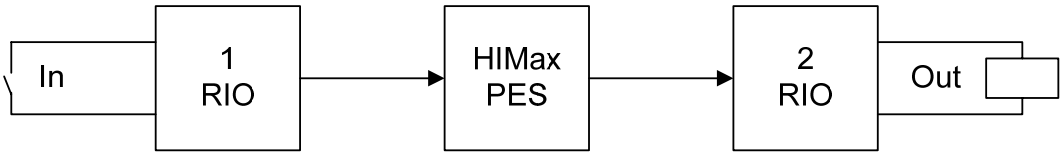


Bild 11: safe**ethernet** Verbindung mit in RIOs

Reaktionszeit mit RIO

$T_R = t_1 + t_2 + t_3 + t_4 + t_5$

T_R Worst Case Reaction Time

t_1 2 * Watchdog-Zeit des 1. RIO

t_2 *ReceiveTMO1*

t_3 2 * Watchdog-Zeit des HIMax PES

t_4 *ReceiveTMO2*

t_5 2 * Watchdog-Zeit des 2. RIO

i Die beiden RIOs 1 und 2 können auch identisch sein. Die Zeiten gelten auch dann, wenn statt einer RIO ein HIMatrix PES eingesetzt wird.

4.7.4 Berechnung der max. Reaktionszeit mit zwei HIMax und einem HIMatrix-PES

Maximale Reaktionszeit T_R („Worst Case“) vom Wechsel eines Gebers (In) im ersten HIMax PES bis zur Reaktion des Ausgangs (Out) im zweiten HIMax PES wie folgt berechnen:

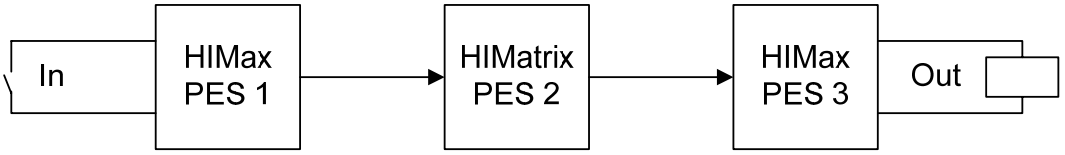


Bild 12: safe**ethernet** Verbindung zweier HIMax mit einer HIMatrix Steuerung

Reaktionszeit mit zwei HIMax Steuerungen und einer HIMatrix Steuerung

$T_R = t_1 + t_2 + t_3 + t_4 + t_5$

T_R Worst Case Reaction Time

t_1 Sicherheitszeit des HIMax PES 1

t_2 *ReceiveTMO1*

t_3 2 * Watchdog-Zeit des HIMatrix PES 2

t_4 *ReceiveTMO2*

t_5 Sicherheitszeit der HIMax PES 3

i Die beiden HIMax PES 1 und 3 können auch identisch sein.
HIMatrix PES 2 kann auch eine HIMax PES sein.

4.7.5 Safeethernet Profile

Safe**ethernet** Profile sind Kombinationen zueinander passender Parameter, die automatisch eingestellt werden, wenn Sie eines der safe**ethernet** Profile auswählen. Für die Parametrierung müssen Sie dann nur die Receive-Timeout und die erwartete Response-Time einzeln konfigurieren.
Das Ziel eines safe**ethernet** Profils besteht darin, den Datendurchsatz im Netzwerk unter Berücksichtigung der physikalischen Gegebenheiten zu optimieren.

Voraussetzung für die Wirksamkeit der Optimierung sind die nachfolgenden Bedingungen:

- Kommunikations-Zeitscheibe ausreichend groß, um in einem CPU-Zyklus alle safeethernet Verbindungen abzuarbeiten.
- wenn mittlere CPU Zykluszeit < Response-Time.
- wenn mittlere CPU Zykluszeit < ProdRate oder ProdRate = 0.

HINWEIS

Störung der safe**ethernet Kommunikation bis hin zum Ausfall möglich!**

Unpassende Kombinationen von CPU-Zyklus, Kommunikations-Zeitscheibe, Response-Time und ProdRate werden bei der Codegenerierung und beim Download/Reload nicht abgelehnt, können aber zu Störungen bei der Kommunikation führen.

Überprüfen Sie in den Control Panels der beiden Steuerungen die Anzeigen *Fehlerhafte Nachrichten* und *Wiederholungen*.

Sechs safe**ethernet** Profile stehen zur Verfügung, aus denen Sie das für die Übertragungsstrecke geeigneten safe**ethernet** Profil auswählen können.

Für sicherheitsgerichtete Prozessdatenkommunikation dürfen nur die Profile "Fast&Noisy", "Medium&Noisy" und "Slow&Noisy" verwendet werden.

Fast & Cleanroom	Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Fast & Noisy	
Medium & Cleanroom	Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Medium & Noisy	
Slow & Cleanroom	Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Slow & Noisy	

4.7.6 Profil I (Fast & Cleanroom)

HINWEIS

Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Für sicherheitsgerichtete Prozessdatenkommunikation dürfen nur die Profile "Fast&Noisy", "Medium&Noisy" und "Slow&Noisy" verwendet werden.

Verwendung

Das Profil *Fast & Cleanroom* ist geeignet für Anwendungen, in idealer Umgebung z.B. Labor!

- Für schnellsten Datendurchsatz
- Für Anwendungen, die eine schnelle Datenübermittlung erfordern
- Für Anwendungen, die eine möglichst geringe Worst Case ReactionTime erfordern

Netzwerkanforderungen

- Fast: 100-Mbit-Technologie (100 Base TX), 1-Gbit-Technologie
- Clean: Störungsfreies Netzwerk.
  Datenverlust durch Netzüberlastung, Einflüsse von außen oder Netzwerkmanipulationen müssen vermieden werden.
- LAN-Switches erforderlich!

Charakteristiken des Kommunikationspfads

- Minimale Verzögerungen
- Erwartete ResponseTime ≤ ReceiveTMO
  (anderenfalls FEHLER bei Parametrierung)

4.7.7 Profil II (Fast & Noisy)

Verwendung

Das Profil *Fast & Noisy* ist das SILworX Standardprofil für die Kommunikation über safe**ethernet**.

- Für schnellen Datendurchsatz
- Für Anwendungen, die eine schnelle Datenübermittlung erfordern
- Für Anwendungen, die eine möglichst geringe Worst Case Reaction Time erfordern

Netzwerkanforderungen

- Fast: 100-Mbit-Technologie (100 Base TX), 1-Gbit-Technologie
- Noisy: Netzwerk ist nicht Störungsfrei.
  Geringe Wahrscheinlichkeit für Verlust von Datenpaketen
  Zeit für ≥ 1 Wiederholung
- LAN-Switches erforderlich!

Charakteristiken des Kommunikationspfads

- Minimale Verzögerungen
- Erwartete ResponseTime ≤ ReceiveTMO / 2
  (anderenfalls FEHLER bei Parametrierung)

4.7.8 Profil III (Medium & Cleanroom)

HINWEIS

Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Für sicherheitsgerichtete Prozessdatenkommunikation dürfen nur die Profile "Fast&Noisy", "Medium&Noisy" und "Slow&Noisy" verwendet werden.

Verwendung
Das Profil *Medium & Cleanroom* ist für Anwendungen in einem Störungsfreien Netzwerk, die eine nur mäßig schnelle Datenübermittlung erfordern.

- Für mittleren Datendurchsatz
- Geeignet für Virtual Private Networks (VPN), in denen der Datenaustausch durch zwischengeschaltete Sicherheitseinrichtungen (Firewalls, Verschlüsselung) langsam, aber fehlerfrei ist.
- Geeignet für Anwendungen, in denen die Worst Case ReactionTime kein kritischer Faktor ist.

Netzwerkanforderungen

- Medium: 10-Mbit- (10 Base T), 100-Mbit- (100 Base TX), 1-Gbit-Technologie
- LAN-Switches erforderlich!
- Clean: Störungsfreies Netzwerk.
  Datenverlust durch Netzüberlastung, Einflüsse von außen oder Netzwerkmanipulationen müssen vermieden werden.
  Zeit für ≥ 0 Wiederholungen.

Charakteristiken des Kommunikationspfads

- Moderate Verzögerungen
- Erwartete ResponseTime ≤ ReceiveTMO (anderenfalls FEHLER bei Parametrierung)

4.7.9 Profil IV (Medium & Noisy)

Verwendung
Das Profil *Medium & Noisy* ist für Anwendungen, die eine nur mäßig schnelle Datenübermittlung erfordern.

- Für mittleren Datendurchsatz
- Für Anwendungen, die nur eine mäßig schnelle Datenübermittlung erfordern
- Geeignet für Anwendungen, in denen die Worst Case ReactionTime kein kritischer Faktor ist

Netzwerkanforderungen

- Medium: 10-Mbit- (10 Base T), 100-Mbit- (100 Base TX), 1-Gbit-Technologie
- LAN-Switches erforderlich!
- Noisy: Netzwerk ist nicht Störungsfrei.
  Geringe Wahrscheinlichkeit für Verlust von Datenpaketen,
  Zeit für ≥ 1 Wiederholung

Charakteristiken des Kommunikationspfads

- Moderate Verzögerungen
- Erwartete ResponseTime ≤ ReceiveTMO / 2 (anderenfalls FEHLER bei Parametrierung)

4.7.10 Profil V (Slow & Cleanroom)

HINWEIS

Für sicherheitsgerichtete Prozessdatenkommunikation nicht geeignet!
Für sicherheitsgerichtete Prozessdatenkommunikation dürfen nur die Profile "Fast&Noisy", "Medium&Noisy" und "Slow&Noisy" verwendet werden.

Verwendung

Das Profil *Slow & Cleanroom* ist für Anwendungen in einem Störungsfreien Netzwerk, die nur eine langsame Datenübermittlung erfordern.

- Für langsamen Datendurchsatz.
- Für Anwendungen, die nur eine langsame Datenübermittlung zu (möglicherweise weit entfernten) Steuerungen erfordern, und dort, wo die Bedingungen der Kommunikationsstrecke nicht Vorhersagbar sind.

Netzwerkanforderungen

- Slow: Datentransfer über ISDN, Standleitung oder Richtfunkverbindung.
- Clean: Störungsfreies Netz.
  Datenverlust durch Netzüberlastung, Einflüsse von außen oder Netzwerkmanipulationen müssen vermieden werden.
  Zeit für ≥ 0 Wiederholungen.

Charakteristiken des Kommunikationspfads

- Moderate Verzögerungen
- Erwartete ResponseTime = ReceiveTMO (anderenfalls FEHLER bei Parametrierung)

4.7.11 Profil VI (Slow & Noisy)

Verwendung

Das Profil *Slow & Noisy* ist für Anwendungen, die nur eine langsame Datenübermittlung zu (möglicherweise weit entfernten) Steuerungen erfordern.

- Für langsamen Datendurchsatz
- Für Anwendungen, Hauptsächlich für Datentransfer über schlechte Telefonleitungen oder gestörte Richtfunkstrecken.

Netzwerkanforderungen

- Slow: Datentransfer über Telefon, Satellit, Funk usw.
- Noisy: Netzwerk ist nicht Störungsfrei.
  Geringe Wahrscheinlichkeit für Verlust von Datenpaketen
  Zeit für ≥ 1 Wiederholung

Charakteristiken des Kommunikationspfads

- Moderate bis lange Verzögerungen
- Erwartete ResponseTime ≤ ReceiveTMO / 2 (anderenfalls FEHLER bei Parametrierung)

4.8 Projektübergreifende Kommunikation

Die projektübergreifende Kommunikation wird benötigt, damit Ressourcen aus verschiedenen Projekten Prozessvariablen miteinander austauschen können.

Die Kommunikation zwischen den beiden Projekten erfolgt über safe**ethernet** und wird im safe**ethernet**-Editor konfiguriert.

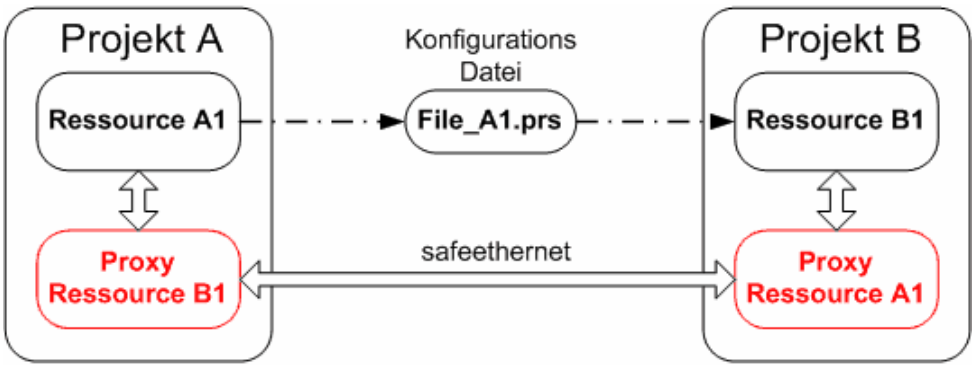


Bild 13: safe**ethernet** Verbindung zwischen Ressource A1 im Projekt A und der Ressource B1 im Projekt B

Als lokales Projekt wird das Projekt bezeichnet, in dem Sie die Konfiguration der safe**ethernet** Verbindung durchführen und die Konfigurationsdatei erstellen.

Als Ziel-Projekt wird das Projekt bezeichnet, in das Sie die Konfigurationsdatei importieren.

Beim Datenaustausch sind das lokale Projekt und das Ziel-Projekt gleichberechtigte Kommunikationspartner.

Die jeweilige Proxy-Ressource dient als Platzhalter für die jeweilige Ressource aus dem externen Projekt und wird für den Import und Export der safe**ethernet** Verbindungen genutzt.

Die *Proxy-Ressource B1* im *Projekt A* ist der Platzhalter der *Ressource B1* aus dem *Projekt B*.

Die *Proxy-Ressource A1* im *Projekt B* ist der Platzhalter der *Ressource A1* aus dem *Projekt A.*

Im lokalen Projekt (hier *Projekt A*) müssen Sie die Proxy-Ressource (hier *Proxy-Ressource B1*) manuell erstellen und konfigurieren. Nach der Konfiguration die Konfigurationsdatei (hier *File_A1.prs*) im Ziel-Projekt (hier von *Ressource B1*) importieren.

Die Konfigurationsdatei *File_A1.prs* enthält die komplette Beschreibung der *Ressource A1* für die safe**ethernet** Verbindung mit der *Ressource B1*. Nach dem Import der Konfigurationsdatei *File_A1.prs* in die *Ressource B1* wird die *Proxy-Ressource A1* automatisch im *Projekt B* angelegt.

4.8.1 Varianten zur projektübergreifenden Kommunikation

In den folgenden beiden Varianten kommunizieren die Projekte A und B über safe**ethernet** miteinander.

Dabei ist in der ersten Variante das Projekt A das lokale Projekt und in der zweiten Variante das Projekt B das lokale Projekt. Grundsätzlich bleibt es dem Anwender überlassen, in welchem der beiden Projekte er die Konfiguration erstellt.

Der Aufwand für beide Wege der Konfiguration ist ungefähr gleich und führt zur gleichen Konfiguration.

Lokales Projekt A

Im lokalen Projekt A konfigurieren Sie die Kommunikation zum Ziel-Projekt B und erstellen die Konfigurationsdateien. Das hat den Vorteil, dass Sie nur die Proxy-Ressource B1 im lokalen Projekt manuell anlegen müssen.

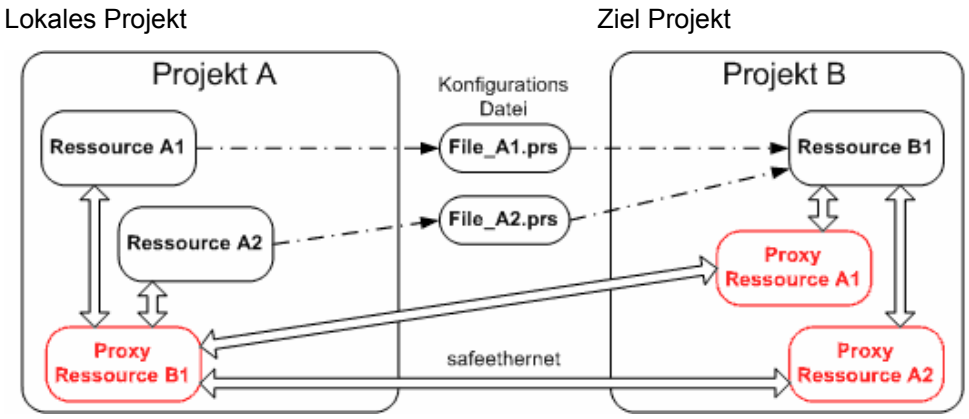


Bild 14: Variante Projekt A als lokales Projekt

Lokales Projekt B

Im lokalen Projekt B konfigurieren Sie die Kommunikation zum Ziel-Projekt A und erstellen die Konfigurations-Files. Das hat den Nachteil, dass Sie zwei Proxy-Ressourcen (A1 und A2) im lokalen Projekt B manuell anlegen müssen.

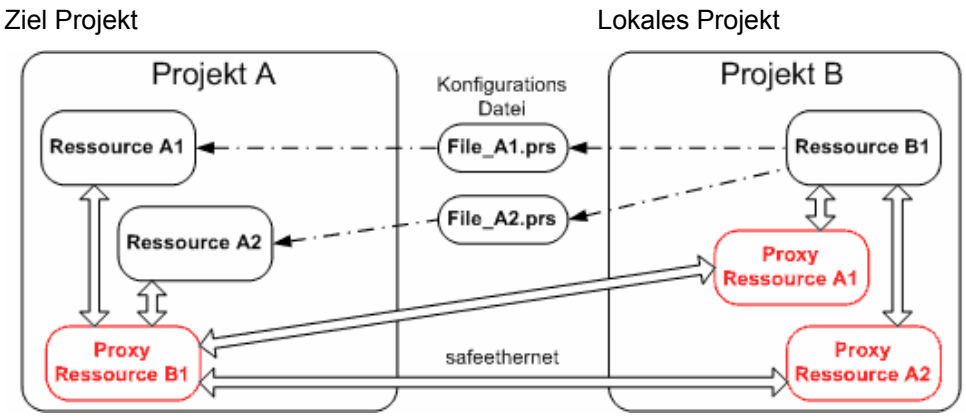


Bild 15: Variante Projekt B als lokales Projekt

4.9 Projektübergreifende Kommunikation zwischen SILworX und ELOP II Factory

In diesem Beispiel wird eine safe**ethernet** Verbindung zwischen HIMax und HIMatrix konfiguriert.

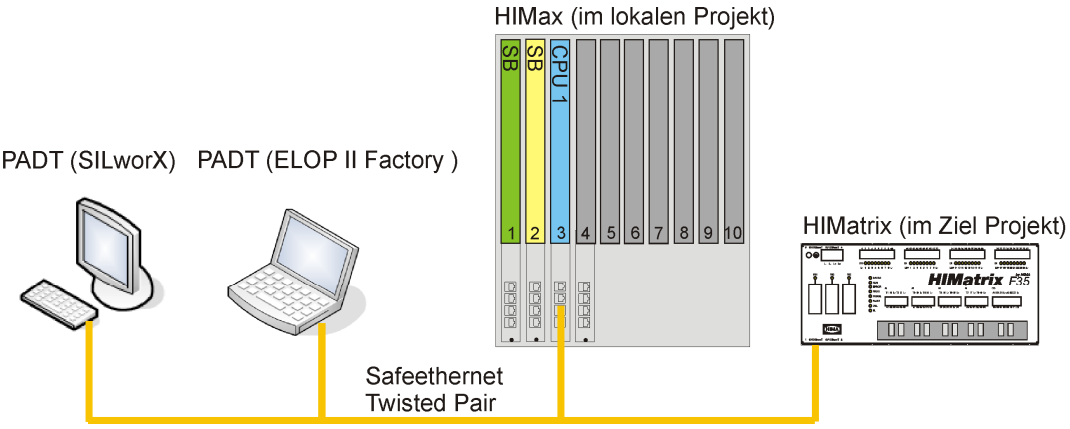


Bild 16: Aufbau zur Kommunikation zwischen SILworX und ELOP II Factory

Ressource des Ziel-Projekts (HIMatrix) öffnen, die im lokalen Projekt (HIMax) als Proxy-Ressource dienen soll.

Folgende Parameter zu dieser Ziel-Ressource ermitteln:

- System-ID
- Sicherheitszeit [ms]
- Watchdog-Zeit [ms]
- IP-Adresse

i Diese Eigenschaften der Ressource sind sicherheitsrelevant und unterliegen Restriktionen. Hinweise dazu finden Sie im Sicherheitshandbuch HI 801 002.

4.9.1 Konfiguration der HIMax im SILworX Projekt

Proxy Ressource erstellen

Eine Proxy-Ressource dient als Platzhalter für eine Ressource in einem externen Projekt und wird für den Import und Export von safe**ethernet** Verbindungen genutzt.

So erstellen Sie im lokalen Projekt eine Proxy-Ressource:

1. Lokales Projekt öffnen, in dem die Proxy-Ressource erstellt werden soll.
2. Im Strukturbaum **Konfiguration** öffnen.
3. Rechtsklicken auf **Konfiguration** und **Neu, Proxy Ressource, ELOP II Factory** wählen.
   ☑ Eine neue Proxy-Ressource wird hinzugefügt.

So konfigurieren Sie im lokalen Projekt eine Proxy-Ressource:

1. Im Kontextmenü der Proxy-Ressource **Eigenschaften** wählen.
2. Im Feld **Name** eindeutigen Namen eintragen.
   Verwenden Sie für die Proxy-Ressource im lokalen Projekt den Namen der Ressource im Ziel-Projekt.
3. Zuvor im Ziel-Projekt ermittelte **System-ID**, **Sicherheitszeit [ms]** und **Watchdog-Zeit [ms]** für diese Proxy-Ressource eintragen.
4. Auf **OK** klicken. Die anderen Parameter können die Standardwerte behalten.

Öffnen Sie die Struktur der Proxy-Ressource:

1. Rechtsklick auf **Hardware** und **Edit**, **HIMatrix Proxy** wählen.
2. Mit **OK** bestätigen, um den Hardware-Editor der Proxy-Ressource zu öffnen.

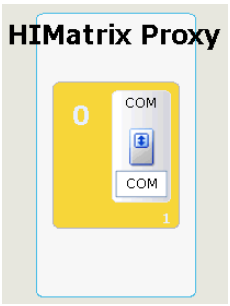


Bild 17: HIMatrix Proxy-Ressource

3. Auf das **COM-Modul** doppelklicken und die ermittelte **IP-Adresse** der Proxy-Ressource eintragen.
4. Auf **Speichern** klicken und dann auf **Schließen**.
5. Diese Schritte für jede weitere Proxy-Ressource im lokalen Projekt wiederholen.

Lokale Ressource mit der Proxy-Ressource verbinden

Erstellen Sie im safe**ethernet**-Editor eine safe**ethernet** Verbindung zwischen der lokalen Ressource und der Proxy-Ressource.

So öffnen Sie den safeethernet Editor der lokalen Ressource:

1. Im Strukturbaum **Konfiguration**, **Ressource** öffnen.
2. Rechtsklicken auf safe**ethernet** und im Kontextmenü **Edit** wählen. In der Objektauswahl befindet sich die angelegte Proxy Ressource.

So erstellen Sie die safeethernet Verbindung zur Proxy Ressource:

1. In der Objektauswahl auf die **Proxy-Ressource** klicken und per Drag&Drop auf eine freie Stelle im Arbeitsbereich des safe**ethernet** Editors ziehen.
2. **Ethernet**-Schnittstellen der lokalen und der Proxy-Ressource auswählen.

Die folgenden Parameter bestimmen den Datendurchsatz sowie die Fehler- und Kollisionstoleranz der safe**ethernet**-Verbindung:

7. **Netzwerk-Profil** (z. B. Fast&Noisy) der safe**ethernet** Verbindung auswählen.
8. **Receive Timeout** und **Response Time** berechnen und eintragen.

Beispiel für Werte der Parameter einer safe**ethernet** Verbindung zur Proxy Ressource:

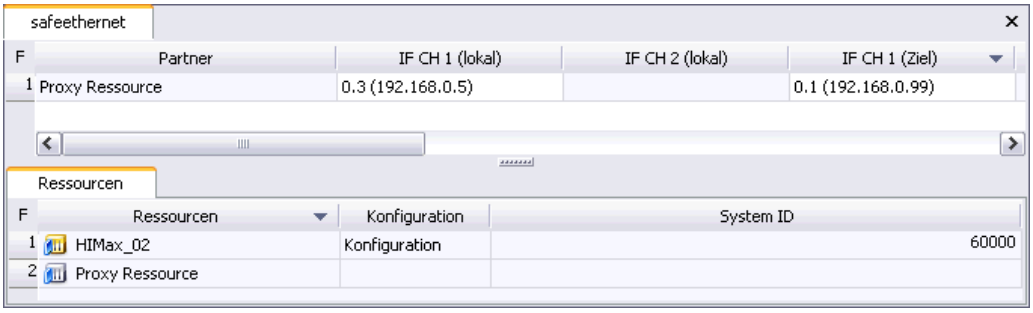


Bild 18: Parameter einer safe**ethernet**-Verbindung zur Proxy-Ressource

Prozessvariablen verbinden

Verbinden Sie die Prozessvariablen in der Detailansicht der safe**ethernet** Verbindung.

So öffnen Sie die Detailansicht einer safeethernet Verbindung:

Voraussetzung: Der safe**ethernet**-Editor der lokalen Ressource muss geöffnet sein.

1. Rechtsklicken auf Zeile **Proxy-Ressource** und Proxy-Ressource öffnen.
2. Im Kontextmenü **Detailansicht** wählen, um die Detailansicht der safe**ethernet** Verbindung zu öffnen.
3. Register **Ressource<->Proxy-Ressource** wählen.

So werden die safeethernet-Sendevariablen hinzugefügt:

Sendevariablen werden von der lokalen Ressource zur Proxy Ressource gesendet.

1. Bereich **Ressource->Proxy-Ressource** wählen**.**
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in die Spalte **Ressource (Ziel)->Ressource (lokal)** ziehen.
3. Diesen Schritt für weitere safe**ethernet** Sendevariablen wiederholen.

So werden die safeethernet-Empfangsvariablen hinzugefügt:

Empfangsvariablen werden von der lokalen Ressource empfangen.

1. Bereich **Ressource<-Proxy-Ressource** wählen.
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in die Spalte **Ressource (Ziel)<-Ressource (lokal)** ziehen.
3. Diesen Schritt für weitere safe**ethernet** Empfangsvariablen wiederholen.

Konfigurationsdatei aus SILworX exportieren

Die in SILworX konfigurierte safe**ethernet**-Verbindung müssen Sie als Konfigurationsdatei mit der Erweiterung ***.prs** exportieren. Diese Konfigurationsdatei kann dann in ELOP II Factory importiert werden, um die safe**ethernet**-Verbindung für die HIMatrix Steuerung zu übernehmen.

So exportieren Sie eine safeethernet Verbindung:

1. Im safe**ethernet** Editor auf **Proxy Ressource** klicken und Kontextmenü öffnen.
2. Im Kontextmenü **Export Verbindung mit Proxy-Ressource** wählen:
   Es öffnet sich ein Standard Dialogfenster zum Speichern einer Datei.
3. Im Dialogfenster Dateinamen für die Konfigurations-Datei eingeben und Datei mit der Erweiterung ***.prs** speichern.
4. Lokales Projekt schließen.

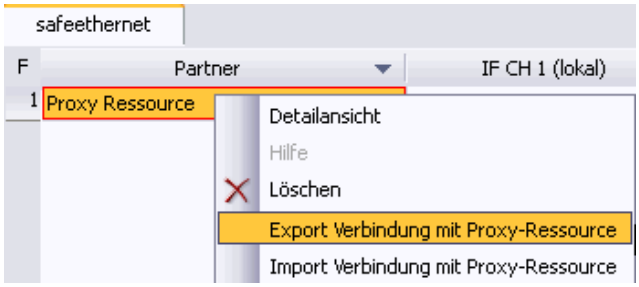


Bild 19: Safe**ethernet**-Verbindung exportieren

Verifikation der safeethernet-Verbindung:

1. Im Strukturbaum **Konfiguration**, **Ressource, safeethernet** öffnen.
2. Auf Schaltfläche **Verifikation** in der **Action Bar** klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen und gegebenenfalls korrigieren.

i Die Konfiguration der safe**ethernet**-Verbindung und das Anwenderprogramm der HIMax Ressource müssen Sie neu compilieren und in die Steuerung laden, damit sie für die Kommunikation der HIMax wirksam werden.

4.9.2 Konfiguration der HIMatrix in ELOP II Factory

So importieren Sie die Konfigurationsdatei in das (HIMatrix) Ziel-Projekt:

1. ELOP II Factory starten.
2. HIMatrix Ziel-Projekt öffnen, in das die Konfigurationsdatei importiert werden soll.
3. Im Strukturbaum Ziel-Ressource wählen und Kontextmenü öffnen.
4. Im Kontextmenü **Import von Verbindungen** wählen:
   Es öffnet sich ein Dialogfenster zum Laden einer Datei mit der Erweiterung ***.prs.**
5. Im Dialogfenster die Konfigurationsdatei wählen, die Sie im lokalen HIMax Projekt erstellt haben, und **OK** klicken.
   ☑ Nach dem Import der Konfigurationsdatei wird die lokale HIMax Ressource automatisch als Proxy-Ressource im HIMatrix Ziel-Projekt angelegt.

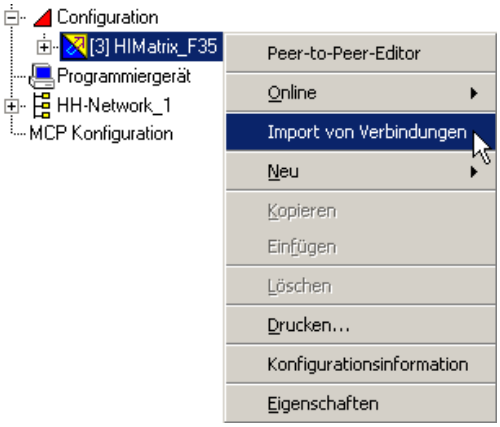


Bild 20: Verbindungen in ELOP II Factory importieren

ELOP II Factory Prozess-Signale zuordnen

Verbinden Sie die Prozess-Signale, in der (HIMatrix) Ziel-Ressource.

Signaleditor über die Menüleiste **Signale**, **Editor** öffnen.

So öffnen Sie in ELOP II Factory den Peer-to-Peer-Editor für die Ziel-Ressource:

1. Im Strukturbaum **Konfiguration, Ressource, Peer-to-Peer-Editor** öffnen.
2. Im **Peer-to-Peer-Editor** das **HH-Netzwerk** für diese Verbindung eintragen.
3. Im **Peer-to-Peer-Editor** auf **Prozess-Signale verbinden** klicken.

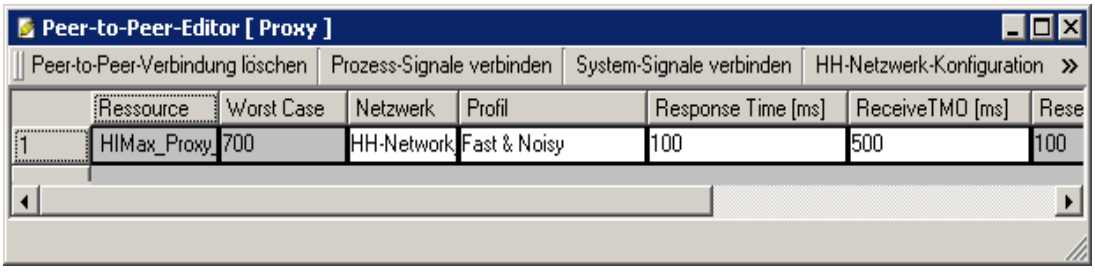


Bild 21: Peer-to-Peer-Editor in ELOP II Factory

i Achten Sie darauf, dass beide Kommunikationspartner das gleiche Profil und die gleichen Einstellungen verwenden (wird beim Import der Konfigurationsdatei automatisch übernommen).

So werden die Peer-to-Peer Sendesignale zugewiesen:

Peer-to-Peer Sendesignale werden von der HIMatrix Ressource zur HIMax Ressource gesendet.

1. Register **HIMatrix Ressource-> HIMax Proxy-Ressource** wählen.
   Das Register enthält die importierten Peer-to-Peer-Sendesignale.
2. Im Signaleditor auf ein **Prozesssignal** klicken und per Drag&Drop auf das Sendesignal im Dialogfenster P2P Prozess-Signale ziehen, welches Sie verbinden wollen.
3. Diesen Schritt für weitere Peer-to-Peer-Sendesignale wiederholen.

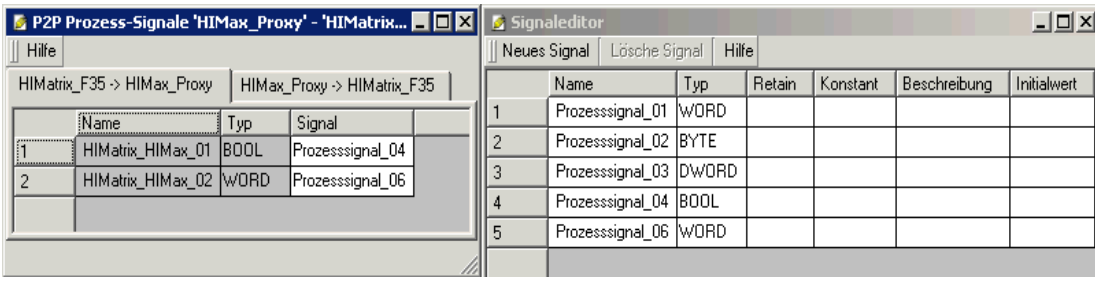


Bild 22: Sendesignale in ELOP II Factory zuweisen

So werden die Peer-to-Peer Empfangssignale zugewiesen:

Peer-to-Peer Empfangssignale werden von der HIMatrix Ressource empfangen.

1. Register **Ressource<-Proxy-Ressource** wählen.
   Das Register enthält die importierten Peer-to-Peer-Empfangssignale.
2. Im Signaleditor auf ein **Prozesssignal** klicken und per Drag&Drop auf das Empfangssignal im Dialogfenster P2P Prozess-Signale ziehen, welches Sie verbinden wollen.
3. Diesen Schritt für weitere Peer-to-Peer-Empfangssignale wiederholen.

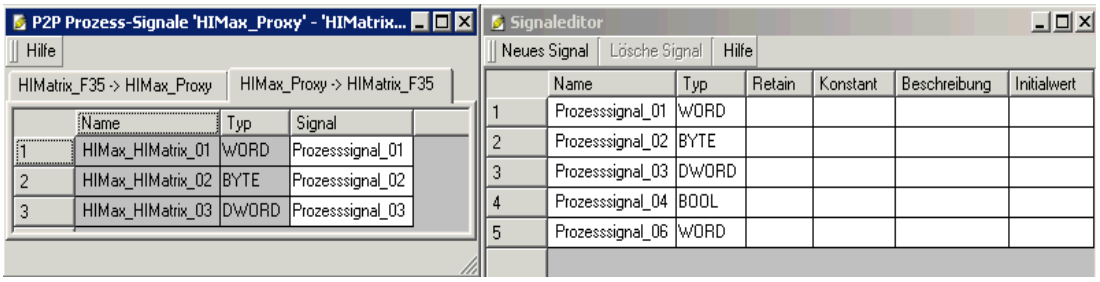


Bild 23: Empfangssignale in ELOP II Factory zuweisen

i Weitere Informationen zum Verbinden von Prozesssignalen in ELOP II Factory in der ELOP II Factory Online Hilfe.

i Die Konfiguration der Peer-to-Peer Verbindung und das Anwenderprogramm der HIMatrix Ressource müssen Sie neu compilieren und in die Steuerung laden. Erst dann wird die Peer-to-Peer Verbindung für die HIMatrix wirksam.

4.10 Control Panel (safeethernet)

Im Control Panel kann der Anwender die Einstellungen der safe**ethernet**-Verbindung überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. Zykluszeit, Bus-Zustand usw.) der safe**ethernet**-Verbindung angezeigt.

So öffnen Sie das Control Panel zur Überwachung der safeethernet Verbindung:

1. Im Strukturbaum **Ressource** wählen.
2. Aus dem Kontextmenü der Ressource **Online** wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels safe**ethernet** wählen.

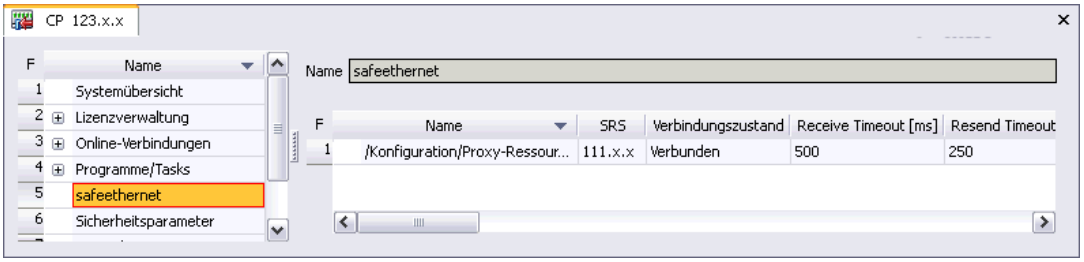


Bild 24: Control Panel zur Verbindungsüberwachung

Statistikwerte zurücksetzen:

Mit der Kontextmenüfunktion können Sie die statistischen Daten (Zykluszeit min, max usw.) auf null zurücksetzen.

So setzen Sie die statistischen Daten der safeethernet Verbindung zurück:

1. Im Strukturbaum safe**ethernet** Verbindung selektieren.
2. Aus dem Kontextmenü der safe**ethernet** Verbindung, safe**ethernet Statistik zurücksetzen** wählen.

4.10.1 Anzeigefeld (safe**ethernet**-Verbindung)

In dem Anzeigefeld werden die folgenden Werte der selektierten safe**ethernet**-Verbindung angezeigt:

Element	Beschreibung
Name	Ressource-Name des Kommunikationspartners
SRS	System.Rack.Slot
Verbindungszustand	Zustand der safe**ethernet**-Verbindung (siehe auch Kapitel 4.4)
Receive-Timeout [ms]	(Siehe auch Kapitel 4.6.2)
Resend-Timeout [ms]	(Siehe auch Kapitel 4.6.5)
Acknowledge Timeout [ms]	(Siehe auch Kapitel 4.6.6)
Min. RspT [ms]	Tatsächliche Response-Time als Minimal-, Maximal-, Letzte- und Durchschnittswert. Siehe auch Kapitel 4.6.3.
Max. RspT [ms]	
Letzte RspT [ms]	
Mittel RspT [ms]	
Fehlerhafte Nachrichten	Anzahl verworfener Nachrichten seit Reset der Statistik.
Wiederholungen	Anzahl der Wiederholungen seit Reset der Statistik.
Anzahl Verbindungserfolge	Anzahl der Verbindungserfolge seit Reset der Statistik.
Early Queue Usage	Anzahl der Nachrichten die in Early Queue gelegt wurden seit Reset der Statistik. Siehe auch Kapitel 4.6.8
Frame-Nr.	Umlaufender Sendungszähler
Ack-Frame-Nr.	Umlaufender Empfangszähler
Monotonie	Umlaufender Nutzdatensendungszähler
Layoutversion	Signatur des aktuellen Kommunikationsendpunkts
Neue Layoutversion	Signatur des neuen Kommunikationsendpunkts
Verbindungssteuerung	Status der Verbindungssteuerung. Siehe auch Kapitel 4.4.
Transport-Steuerung Kanal 1	Freigabe von Transportweg Kanal1. Siehe auch Kapitel 4.4.
Transport-Steuerung Kanal 2	Freigabe von Transportweg Kanal2. Siehe auch Kapitel 4.4.
Qualität Kanal1	Zustand von Transportweg Kanal1. Siehe auch Kapitel 4.4.
Qualität Kanal2	Zustand von Transportweg Kanal2. Siehe auch Kapitel 4.4.
Spät erhaltene redundante Nachrichten	Bei redundanten Transportwegen. Anzahl der verspätet empfangenen Nachrichten seit Reset der Statistik.
Verlorene redundante Nachrichten	Bei redundanten Transportwegen. Anzahl der auf nur einem der beiden Transportwege empfangenen Nachrichten seit Reset der Statistik.
Protokollversion	0 und 1 = Alte Protokoll-Version für HIMatrix < V7 2 = Neue Protokoll Version für HIMax

Tabelle 29: Anzeigefeld der safe**ethernet**-Verbindung

5 PROFINET-IO

PROFINET-IO ist das auf Ethernet-Technologie basierte Übertragungsprotokoll der PROFIBUS Nutzerorganisation für die Automatisierung.

Wie bei PROFIBUS-DP werden die dezentralen Feldgeräte bei PROFINET-IO über eine Gerätebeschreibung (GSDML-Datei) in SILworX eingebunden.

Der HIMA PROFINET-IO Controller entspricht der Conformance Class A und unterstützt die Non-Real-Time (NRT)- und die Real Time (RT) Kommunikation zu den PROFINET-IO Devices. Wobei die Real-Time Kommunikation für den zeitkritischen Datenaustausch und die Non-Real Kommunikation für nicht zeitkritische Vorgänge (z.B. azyklisches Lesen/Schreiben) automatisch verwendet werden.

Eine redundante PROFINET-IO Verbindung kann nur durch die Konfiguration eines zweiten PROFINET-IO Controller/Device und Anpassungen im Anwenderprogramm erreicht werden.

5.1 PROFINET-IO Funktionsbausteine

Zum azyklischen Datenaustausch stehen Ihnen in SILworX die funktional gleichen Funktionsbausteine wie bei PROFIBUS-DP zur Verfügung. Mit diesen PROFINET-IO Funktionsbausteinen können Sie den HIMA PROFINET-IO Controller und die ihm zugeordneten PROFINET-IO Devices optimal den Erfordernissen Ihres Projekts anpassen.

Folgende PROFINET-IO Funktionsbausteine stehen zur Verfügung:

Funktionsbaustein	Beschreibung der Funktion
MSTAT 6.9.1	Zustand des Controllers durch das Anwenderprogramm steuern
RALRM 6.9.2	Alarmmeldungen der Devices lesen
RDREC 6.9.4	Azyklische Datensätze der Devices lesen
SLACT 6.9.5	Zustand der Devices durch das Anwenderprogramm steuern
WRREC 6.9.6	Azyklische Datensätze der Devices schreiben

Tabelle 30: Übersicht PROFINET-IO Funktionsbausteine

Die PROFINET-IO Funktionsbausteine parametrieren Sie wie die PROFIBUS-DP Funktionsbausteine, siehe ab Kapitel 6.9.

5.2 HIMA PROFINET-IO Controller

Dieses Kapitel beschreibt die Eigenschaften des HIMA PROFINET-IO Controller sowie die Menüfunktionen und Dialoge in SILworX, die Sie zur Konfiguration des HIMA PROFINET-IO Controllers benötigen.

5.3 Systemanforderung

Benötigte Ausstattung und Systemanforderung:

Element	Beschreibung
Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für PROFINET-IO nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT.
Aktivierung	Die Freischaltung erfolgt per Software-Freischaltcode, siehe Kapitel 3.5.

Tabelle 31: Systemanforderung und Ausstattung für PROFINET-IO Controller

PROFINET-IO Controller Eigenschaften:

Element	Beschreibung
Sicherheitsgerichtet	Nein
Übertragungsrate	100 Mbit/s vollduplex
Transportweg	Ethernet-Schnittstellen der COM-Module Verwendete Ethernet-Schnittstellen simultan auch für andere Protokolle nutzbar.
Konformitätsklasse	Der PROFINET-IO Controller entspricht den Anforderungen der Conformance Class A.
Real Time Class	RT-Klasse 1
Max. Anzahl PROFINET-IO Controller	Für jedes COM-Modul können Sie einen PROFINET-IO Controller konfigurieren.
Max. Anzahl PROFINET-IO Devices Applikationsbeziehungen (AR's)	Ein PROFINET-IO Controller kann mit max. 64 PROFINET-IO Devices eine Applikationsbeziehung (AR) aufbauen.
Anzahl (CR's pro AR) Kommunikationsbeziehungen	Standard: 1 Input CR, 1 Output CR, 1 Alarm CR
Max. Prozessdatenlänge einer Kommunikationsbeziehung (CR)	Output: max 1440 Bytes Input: max. 1440 Bytes (RT-Frame over UDP)
Priorisierung der Daten	Möglich über die Einstellung der *Reduction Rate* auf der Device Ebene.

Tabelle 32: Eigenschaften PROFINET-IO Controller

5.4 PROFINET-IO Beispiel

Hier erfahren Sie, wie Sie den HIMA PROFINET-IO Controller mit einem beliebigen PROFINET-IO Device verbinden.

5.4.1 Anlegen des HIMA PROFINET-IO Controller in SILworX

So legen Sie einen neuen HIMA PROFINET-IO Controller an:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFINET-IO Controller** wählen, um einen neue PROFINET-IO Controller hinzuzufügen.
3. Im Kontextmenü vom PROFINET-IO Controller **Eigenschaften** wählen.
4. **COM Modul** auswählen.

Konfiguration des PROFINET-IO Device mit SILworX

So legen Sie ein HIMax PROFINET-IO Device im PROFINET-IO Controller an:

1. Im Kontextmenü des PROFINET-IO Controller **Neu**, **PROFINET-IO Device** wählen.

So lesen Sie die GSDML-Bibliotheksdatei aus einer externen Datenquelle ein (z.B. CD, USB-Stick, Internet):

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **GSDML-Bibliothek** wählen.
2. Im Kontextmenü der GSDML-Bibliothek**, GSDML Datei hinzufügen** wählen und die zum PROFINET-IO Device zugehörige GSDML-Datei einlesen.

---

i Die GSDML-Bibliotheksdatei beinhaltet meistens mehrere Geräte eines Herstellers.

So laden Sie die GSDML-Datei für ein neues PROFINET-IO Device:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **PROFINET-IO Device** wählen.
2. Im Kontextmenü **Eigenschaften** wählen und das Register Parameter öffnen.
3. Im Dropdown-Menü **GSDML-Datei**, die zum PROFINET-IO Device zugehörige GSDML-Bibliotheksdatei auswählen und **Eigenschaften** schließen.

So wählen Sie den Data Access Point (DAP) für das PROFINET-IO Device:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **PROFINET-IO Device, DAP Modul** wählen.
2. Im Kontextmenü **Device Access Point (DAP) auswählen** wählen und den passenden Datensatz zum PROFINET-IO Device auswählen.

PROFINET-IO Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, PROFINET-IO Controller** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls Fehler korrigieren.

Ermitteln des PROFINET-IO Devices im Netzwerk

So finden Sie im Ethernet Netzwerk das PROFINET-IO Device:

1. Modul-Login auf das Kommunikationsmodul mit dem **PROFINET-IO Controller** ausführen.
2. Im Strukturbaum der Online-Ansicht **PROFINET-IO-Controller, PROFINET-IO Netzwerkteilnehmer** wählen.
3. Im Kontektmenü **PROFINET-Netzwerkteilnehmer ermitteln** wählen.
   ☑ Alle PROFINET-Devices im Netzwerk dieses PROFINET-IO Controllers werden aufgelistet.

So konfigurieren Sie das PROFINET-IO Device in der Online-Ansicht:

1. In der Geräteliste, Rechtsklick auf das zu konfigurierende PROFINET-IO Device, um die Einstellungen zu ändern.
2. Mit der Kontextmenüfunktion **Setze PROFINET-IO Gerätenamen**, setzen Sie den Gerätenamen.
   ☑ Stellen Sie sicher, dass der PROFINET-IO Gerätename mit der Projektierung übereinstimmt. Groß-/Kleinschreibung beachten!
3. Mit der Kontextmenüfunktion **Netzwerkeinstellungen** setzen Sie die IP-Adresse, Subnet Maske und den Gateway.

i Die Netzwerkeinstellungen des PROFINET-IO Device müssen im PROFINET-IO Controller in SILworX eingestellt sein, sonst ist eine Kommunikation nicht möglich.

5.5 Menüfunktion im PROFINET-IO Controller

5.5.1 Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des PROFINET-IO Controller öffnet den Dialog **Eigenschaften**.

Element	Beschreibung
Typ	PROFINET-IO Controller
Name	Beliebiger, eindeutiger Name, für einen PROFINET-IO Controller
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und CPU ausgetauscht werden. Ist die *Refresh Rate* Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 4 ...(2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert
Modul	Auswahl des COM-Moduls auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld *Max. CPU Load [%]* übernehmen. Deaktiviert: Kein Limit der CPU-Last, für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
RPC Port Server	Remote Procedure Call Port Wertebereich: 1024 ... 65535 Standardwert: 49152 RPC Port Server und RPC Port Client dürfen nicht identisch sein!
RPC Port Client	Remote Procedure Call Port Wertebereich: 1024 ... 65535 Standardwert: 49153 RPC Port Server und RPC Port Client dürfen nicht identisch sein!

Tabelle 33: Allgemeine Eigenschaften des PROFINET-IO Controller

5.6 Menüfunktionen PROFINET-IO Device (im Controller)

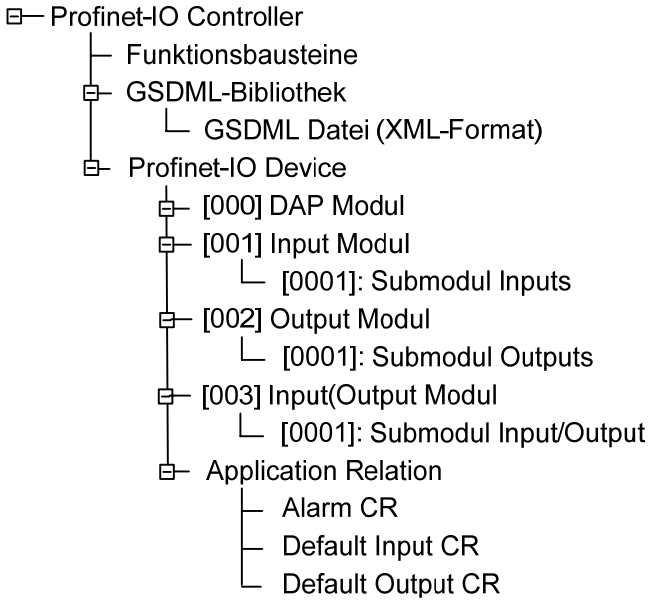


Bild 25: Strukturbaum des PROFINET-IO Controllers

5.6.1 Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des PROFINET-IO Device öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Beliebiger, eindeutiger Name, für das PROFINET-IO Device
Steckplatz	Wertebereich: 0 ... 125 Standardwert: 0
IP-Adresse	Ziel IP-Adresse des Kommunikationspartners (bei direktem Host-Routing) oder Netzwerkadresse (bei Subnet-Routing). Wertebereich: 0.0.0.0 ... 255.255.255.255 Standardwert: 0.0.0.0 Verwenden Sie keine belegten IP-Adressen, siehe Kapitel 3.6.3.
Subnet-Maske	Definiert Ziel-Adressbereich für einen Routing-Eintrag. 255.255.255.255 (bei direktem Host-Routing) oder Subnet-Maske des adressierten Subnetzes. Wertebereich: 0.0.0.0 ... 255.255.255.255 Standardwert: 255.255.255.255
GSDML-Datei	GSDML ist die Abkürzung für Generic Station Description Markup Language und ist eine XML basierte Beschreibungssprache. Die GSDML Datei enthält die Stammdaten des PROFINET Device

Tabelle 34: Register Parameter des PROFINET-IO Device

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung und die Ausstattung der Input/Output PROFINET-IO Module, die selbsterklärend sind.

5.6.2 DAP Modul (Device Access Point Modul)

Unterhalb eines PROFINET Devices wird immer ein Modul für die Busanschaltung (DAP: Device Access Point Modul) angelegt. Das DAP Modul ist Standard und kann nicht gelöscht werden.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des DAP Moduls öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Beliebiger, eindeutiger Name, für das DAP Modul
Steckplatz	Nicht änderbar. Standardwert: 0

Tabelle 35: Register Parameter im Eigenschaftendialog des DAP Moduls

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung des DAP-Moduls, die selbsterklärend sind.

5.6.3 Input/Output PROFINET-IO Module

In den PROFINET-IO Input Modulen werden die Eingangsvariablen des HIMax PROFINET-IO Controllers eingetragen, die vom PROFINET-IO Device gesendet werden.

In den PROFINET-IO Output Modulen werden die Ausgangsvariablen des HIMax PROFINET-IO Controllers eingetragen, die zum PROFINET-IO Device gesendet werden.

So legen Sie die benötigten PROFINET-IO Module an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Device** öffnen.
2. Im Kontextmenü von PROFINET-IO Device **Neu** wählen.
3. Passende Module wählen.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü der Input/Output PROFINET-IO Module öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Name der Input/Output PROFINET-IO Module
Steckplatz	0 bis 32767 Standardwert: 1

Tabelle 36: Register Parameter der I/O PROFINET-IO Module

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung und die Ausstattung der Input/Output PROFINET-IO Module, die selbsterklärend sind.

5.6.4 Submodul Input

Mit den Parametern der Submodule werden die Kommunikationsbeziehungen und das Verhalten der Module bei Verbindungsunterbrechung eingestellt.

Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü der Input Submodule öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Name des Input Submoduls
Substeckplatz	Nicht änderbar Standardwert: 1
IO Data CR, Eingänge	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in der die Eingänge des Submoduls übertragen werden sollen. - keine - Default Input CR
Eingangsdaten vom Controller akzeptiert	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in welcher der IO Consumer Status (CS) des Submoduls übertragen werden soll. - keine - Default Output CR
Shared Input	Aktiviert: Mehrere PROFINET-IO Controller können auf die Eingänge zugreifen. Deaktiviert: Nur ein PROFINET-IO Controller kann auf die Eingänge zugreifen.
Input-Werte bei IO-CR Unterbrechung	Verhalten der Input Variablen dieses PROFINET-IO Submodul bei Verbindungsunterbrechung. Letzten Wert behalten: Die Input Variablen werden auf dem momentanen Wert eingefroren und bis zur erneuten Verbindungsaufnahme verwendet. Initialwerte annehmen: Für die Input Variablen werden die Initialdaten verwendet.

Tabelle 37: Eigenschaftendialog von Submodul Input

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung und die Ausstattung des Input Submoduls, die selbsterklärend sind.

Edit

Die Menüfunktion **Edit** aus dem Kontextmenü der Input Submodule öffnet den Dialog **Edit**. Das Dialogfenster enthält die folgenden Register:

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFINET-IO Submoduls im Anwenderprogramm auszuwerten.

Element	Beschreibung
Eingangsdaten gültig	True Eingangsdaten gültig GOOD False Eingangsdaten ungültig BAD_BY_DEVICE, BAD_BY_CONTROLLER
Eingangsdaten vom Controller akzeptiert	True Eingangsdaten gültig GOOD False Eingangsdaten ungültig BAD_BY_SUBSLOT, BAD_BY_SLOT, BAD_BY_DEVICE, BAD_BY_CONTROLLER.

Tabelle 38: Edit Dialog von Submodul Input

Im Register **Prozessvariablen** werden die Eingangsvariablen eingetragen.

5.6.5 Submodul Output

Mit den Parametern der Submodule werden die Kommunikationsbeziehungen und das Verhalten der Module bei Verbindungsunterbrechung eingestellt.

Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü der Output Submodule öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Name des Output Submoduls
Substeckplatz	Nicht änderbar Standardwert: 1
IO Data CR, Ausgänge	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in der die Ausgänge des Submoduls übertragen werden sollen. - keine - Default Input CR
Ausgangsdaten vom Device akzeptiert	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in welcher der IO Consumer Status (CS) des Submoduls übertragen werden soll. - keine - Default Output CR

Tabelle 39: Eigenschaftendialog von Submodul Input

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung und die Ausstattung des Input Submoduls, die selbsterklärend sind.

Edit

Die Menüfunktion **Edit** aus dem Kontextmenü des Submoduls Output öffnet den Dialog **Edit**. Das Dialogfenster enthält die folgenden Register:

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFINET-IO Submoduls im Anwenderprogramm auszuwerten.

Element	Beschreibung	
Ausgangsdaten gültig	True	Ausgangsdaten gültig GOOD
	False	Ausgangsdaten ungültig BAD_BY_DEVICE, BAD_BY_CONTROLLER
Ausgangsdaten vom Controller akzeptiert	True	Ausgangsdaten gültig GOOD
	False	Ausgangsdaten ungültig BAD_BY_SUBSLOT, BAD_BY_SLOT, BAD_BY_DEVICE, BAD_BY_CONTROLLER.

Tabelle 40: Edit Dialog von Submodul Output

Im Register **Prozessvariablen** werden die Ausgangsvariablen eingetragen

5.6.6 Submodul Inputs und Outputs

Mit den Parametern der Submodule werden die Kommunikationsbeziehungen und das Verhalten der Module bei Verbindungsunterbrechung eingestellt.

Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü der Input/Output Submodule öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Name des Input/Output Submoduls
Substeckplatz	Nicht änderbar Standardwert: 1
IO Data CR, Eingänge	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in der die Eingänge des Submoduls übertragen werden sollen. - keine - Default Input CR
IO Data CR, Ausgänge	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in der die Ausgänge des Submoduls übertragen werden sollen. - keine - Default Output CR
Eingangsdaten vom Controller akzeptiert	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in welcher der IO Consumer Status (CS) des Submoduls übertragen werden soll. - keine - Default Output CR
Ausgangsdaten vom Device akzeptiert	Auswahl der Kommunikationsbeziehung (Communication Relation CR) in welcher der IO Consumer Status (CS) des Submoduls übertragen werden soll. - keine - Default Input CR
Input-Werte bei IO-CR Unterbrechung	- Letzten Wert behalten - Initialwerte annehmen

Tabelle 41: Eigenschaftendialog von Submodul Input/Output

Die Register **Modell** und **Features** enthalten eine weitere Beschreibung und die Ausstattung des Input und Output Submoduls, die selbsterklärend sind.

Edit

Die Menüfunktion **Edit** aus dem Kontextmenü der Submodule Input/Output öffnet den Dialog **Edit**. Das Dialogfenster enthält die folgenden Register:

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFINET-IO Submoduls im Anwenderprogramm auszuwerten.

Element	Beschreibung	
Ausgangsdaten gültig	True	Ausgangsdaten gültig GOOD
	False	Ausgangsdaten ungültig BAD_BY_DEVICE, BAD_BY_CONTROLLER
Ausgangsdaten vom Controller akzeptiert	True	Ausgangsdaten gültig GOOD
	False	Ausgangsdaten ungültig BAD_BY_SUBSLOT, BAD_BY_SLOT, BAD_BY_DEVICE, BAD_BY_CONTROLLER.
Eingangsdaten gültig	True	Eingangsdaten gültig GOOD
	False	Eingangsdaten ungültig BAD_BY_DEVICE, BAD_BY_CONTROLLER
Eingangsdaten vom Controller akzeptiert	True	Eingangsdaten gültig GOOD
	False	Eingangsdaten ungültig BAD_BY_SUBSLOT, BAD_BY_SLOT, BAD_BY_DEVICE, BAD_BY_CONTROLLER.

Tabelle 42: Edit Dialog von Submodul Input und Output

Im Register **Prozessvariablen** werden die Eingangsvariablen und Ausgangsvariablen im jeweiligen Bereich eingetragen

5.6.7 Application Relation

Eine Applikationsbeziehung (AR: Application Relation) ist ein logisches Gebilde zum Datenaustausch zwischen Controller und Device. Die Datenübertragung findet innerhalb der Applikationsbeziehung über eine oder bis zu fünf Kommunikationsbeziehungen (CR: Communication Relation) statt.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü von Application Relation öffnet den Dialog **Eigenschaften**.

Element	Beschreibung
Name	Nicht änderbar
UDP RT Port Controller	UDP Port des Controller Nicht änderbar
AR UUID	Kennzahl zur eindeutigen identifizierung der Applikationsbeziehung (AR). Nicht änderbar
Verbindungsaufbau Timeout Faktor	Mit diesem Parameter wird die Zeit berechnet, welche aus Sicht eines PROFINET-IO-Devices während des Verbindungsaufbaus zwischen dem Senden der Antwort auf den Connect-Request und dem Empfang eines neuen Requests vom PROFINET-IO-Controller maximal verstreichen darf. Wertebereich: 1 - 1000 (x 100 ms) Standardwert: 600
Supervisor darf die AR übernehmen	Definition, ob ein PROFINET-IO Supervisor die Applikationsbeziehung (AR) übernehmen darf. Ältere Devices verschiedener Hersteller benötigen diese Einstellung. 0 Nicht erlaubt 1 Erlaubt Standardwert: 0

Tabelle 43: Eigenschaftendialog von Application Relation

5.6.8 Alarm CR

Innerhalb einer Applikationsbeziehung können mehrere Kommunikationsbeziehungen (CR's: Communication Relations) aufgebaut werden.

Über die Kommunikationsbeziehung Alarm CR überträgt ein PROFINET-IO Device alarme an den PROFINET-IO Controller.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü von Application Relation öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Parameter:

Element	Beschreibung
Name	Nicht änderbar
VLAN-ID, hochpriore Alarme	Jedem Virtuellen LAN (VLAN) wird zur Abschottung eine eindeutige Nummer zugeordnet. Ein Gerät, das zum VLAN mit der ID=1 gehört, kann mit jedem anderen Gerät im gleichen VLAN kommunizieren, nicht jedoch mit einem Gerät in einem anderen VLAN mit ID=2, 3, ... Wertebereich siehe auch IEC-61158-6 0x000 Kein VLAN 0x001 Standard VLAN 0x002 bis 0xFFF Siehe IEEE 802.1 Q Standardwert: 0
VLAN-ID, niederpriore Alarme	Beschreibung, siehe VLAN-ID, hochpriore Alarme Standardwert: 0
Alarm Priorität	Benutzer Priorität verwenden: Die vom Benutzer vergebene Priorität wird verwendet. Benutzer Priorität ignorieren: Die vom Benutzer vergebene Priorität wird ignoriert. Der generierte Alarm ist immer niedrigprior.
Alarm Sendeversuche	Maximale Anzahl an Sendeversuchen des Devices, falls der Controller nicht antwortet. Wertebereich 3 bis 15 Standardwert: 10
Alarm Timeout Faktor	Mit dem RTA Timeout Factor wird die Timeoutzeit des Devices berechnet, welche nach dem Versenden eines RTA_Data(Alarm)-Frames und dem Erhalt des RTA_ACK-Frames maximal verstreichen darf . RTA Timeout = RTA Timeout Factor x 100 ms Wertebereich 1 bis 65535 Standardwert: 5

Tabelle 44: Eigenschaftendialog von Alarm CR

5.6.9 Input CR

Innerhalb einer Applikationsbeziehung können mehrere Kommunikationsbeziehungen (CR's: Communication Relations) aufgebaut werden.

Über die Kommunikationsbeziehung Input CR überträgt ein PROFINET-IO Device Variablen zum PROFINET-IO Controller.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü von Default Input CR öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Parameter:

Element	Beschreibung
Name	Beliebiger, eindeutiger Name, für ein Input CR Das Default Input CR ist nicht änderbar
Typ	1 (Nicht änderbar)
Send Clock Faktor	Der Send Clock Faktor bestimmt den Send Clock der zyklischen Datenübertragung einer IO-CR. **Send Clock = Send Clock Faktor x 31,25 µs** Wertebereich: 1 bis 128 Standardwert: 32
Reduktions Faktor	Der Reduktions Faktor ermöglicht die Untersetzung der tatsächlichen Zykluszeit des Sendens der Daten einer IO-CR. Die tatsächliche Zykluszeit der Daten errechnet sich aus: **Send Cycle = Reduktions Faktor x Send Clock** Wertebereich: 1 bis 16384 Standardwert: 32
Watchdog Faktor	Mit dem Watchdog Factor wird diejenige Zeit berechnet, welche aus Sicht eines IO-CR-Consumers zwischen dem Empfang zweier Frames maximal verstreichen darf: **Watchdog Time = WatchdogFactor x Send Clock Factor x ReductionRatio x 31,25 µs** Wertebereich: 1 bis 7680 Standardwert: 3
VLAN-ID	Jedem Virtuellen LAN (VLAN) wird zur Abschottung eine eindeutige Nummer zugeordnet. Ein Gerät, das zum VLAN mit der ID=1 gehört, kann mit jedem anderen Gerät im gleichen VLAN kommunizieren, nicht jedoch mit einem Gerät in einem anderen VLAN mit ID=2, 3, ... Wertebereich siehe auch IEC-61158-6 0x000 Kein VLAN 0x001 Standard VLAN 0x002 bis 0xFFF Siehe IEEE 802.1 Q Standardwert: 0

Tabelle 45: Eigenschaftendialog von Default Input CR

Die Menüfunktion **Edit** aus dem Kontextmenü von Default Input CR öffnet den Dialog **Systemvariablen**, und enthält die folgende Systemvariable.

Element	Beschreibung
Daten Status Input CR	

Tabelle 46: Dialog Edit von Default Input CR

5.6.10 Output CR

Innerhalb einer Applikationsbeziehung können mehrere Kommunikationsbeziehungen (CR's: Communication Relations) aufgebaut werden.

Über die Kommunikationsbeziehung Output CR überträgt der PROFINET-IO Controller Variablen zum PROFINET-IO Device.

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü von Output CR öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Parameter:

Element	Beschreibung
Name	Beliebiger, eindeutiger Name, für ein OUTPUT CR Das Default Output CR ist nicht änderbar
Typ	2 (Nicht änderbar)
Send Clock Faktor	Der Send Clock Faktor bestimmt den Send Clock der zyklischen Datenübertragung einer IO-CR. **Send Clock = Send Clock Faktor x 31,25 µs** Wertebereich: 1 bis 128 Standardwert: 32
Reduktions Faktor	Zur Einstellung der Übertragungshäufigkeit. Der Reduktions Faktor ermöglicht die Untersetzung der tatsächlichen Zykluszeit des Sendens der Daten einer IO-CR. Die tatsächliche Zykluszeit der Daten errechnet sich aus: **Send Cycle = Reduktions Faktor x Send Clock** Wertebereich: 1 bis 16384 Standardwert: 32
Watchdog Faktor	Mit dem Watchdog Factor wird diejenige Zeit berechnet, welche aus Sicht eines IO-CR-Consumers zwischen dem Empfang zweier Frames maximal verstreichen darf: **Watchdog Time = WatchdogFactor x Send Clock Factor x ReductionRatio x 31,25 µs** Wertebereich: 1 bis 7680 Standardwert: 3
VLAN-ID	Jedem Virtuellen LAN (VLAN) wird zur Abschottung eine eindeutige Nummer zugeordnet. Ein Gerät, das zum VLAN mit der ID=1 gehört, kann mit jedem anderen Gerät im gleichen VLAN kommunizieren, nicht jedoch mit einem Gerät in einem anderen VLAN mit ID=2, 3, ... Wertebereich siehe auch IEC-61158-6 0x000 Kein VLAN 0x001 Standard VLAN 0x002 bis 0xFFF Siehe IEEE 802.1 Q Standardwert: 0

Tabelle 47: Eigenschaftendialog von Default Output CR

5.7 HIMA PROFINET-IO Device

Dieses Kapitel beschreibt die Eigenschaften des HIMA PROFINET-IO Device sowie die Menüfunktionen und Dialoge in SILworX, die Sie zur Konfiguration des HIMA PROFINET-IO Controllers benötigen.

5.8 Systemanforderung

Benötigte Ausstattung und Systemanforderung:

Element	Beschreibung
Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für PROFINET-IO nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT.
Aktivierung	Die Freischaltung erfolgt per Software-Freischaltcode, siehe Kapitel 3.5.

Tabelle 48: Systemanforderung und Ausstattung für PROFINET-IO Controller

PROFINET-IO Device Eigenschaften:

Element	Beschreibung
Sicherheitsgerichtet	Nein
Übertragungsrate	100 Mbit/s vollduplex
Transportweg	Ethernet-Schnittstellen der COM-Module Verwendete Ethernet-Schnittstellen simultan auch für andere Protokolle nutzbar.
Konformitätsklasse	Das PROFINET-IO Device entspricht den Anforderungen der Conformance Class A.
Real Time Class	RT-Klasse 1
Max. Anzahl PROFINET-IO Devices	Für jedes COM-Modul können Sie ein PROFINET-IO Device konfigurieren.
Max. Anzahl Applikationsbeziehungen (AR's) zum PROFINET-IO Controller	Ein PROFINET-IO Device kann max. 1 Applikationsbeziehungen (AR's) zu einem PROFINET-IO Controllern aufbauen.
Max. Anzahl (CR's pro AR) Kommunikationsbeziehungen	Standard: 1 Input, 1 Output, 1 Alarm
Max. Prozessdatenlänge aller konfigurierten PROFINET-IO Module	Output: max. 1440 Bytes Input: max. 1440 Bytes
Priorisierung der Daten	Möglich über die Einstellung der *Reduction Rate* auf der Device Ebene.

Tabelle 49: Eigenschaften PROFINET-IO Controller

5.9 PROFINET-IO Beispiel

In diesem Beispiel tauscht ein HIMA PROFINET-IO Controller Variablen mit einem HIMA PROFINET-IO Device aus. Hier erfahren Sie, wie Sie den HIMA PROFINET-IO Controller und den HIMA PROFINET-IO Device anlegen und parametrieren.

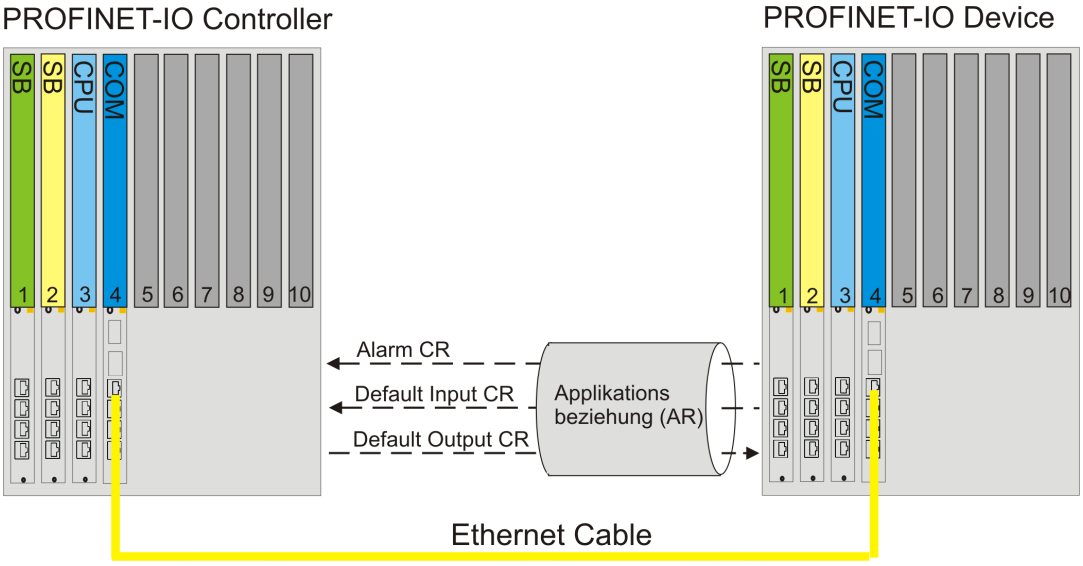


Bild 26: Kommunikation über PROFINET IO

Die Kommunikationsmodule der beiden HIMax Steuerungen werden über die Ethernetschnittstelle mit einem Ethernet Kabel verbunden.

Eine Applikationsbeziehung (AR: Application Relation) ist ein logisches Gebilde zum Datenaustausch zwischen Controller und Device. Die Datenübertragung innerhalb der Applikationsbeziehung findet in diesem Beispiel über die Standard Kommunikationsbeziehungen (Alarm CR, Default Input CR und Default Output CR) statt. Diese Kommunikationsbeziehungen sind bereits standardmäßig in den Input und Output Modulen eingestellt.

Folgende Globalen Variablen müssen Sie für dieses Beispiel in SILworX anlegen:

Globale Variable	Typ
PN_Device_Controller1	UINT
PN_Device_Controller2	DWORD
PN_Device_Controller3	DWORD
PN_Device_Controller4	BYTE
PN_Controller_Device1	UINT
PN_Controller_Device2	BYTE

5.9.1 Konfiguration des PROFINET-IO Device in SILworX

So legen Sie ein neues HIMA PROFINET-IO Device an:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFINET-IO Device** wählen, um ein neues PROFINET-IO Device hinzuzufügen.
3. Im Kontextmenü vom PROFINET-IO Controller **Eigenschaften** wählen.
4. **COM Modul** auswählen.

So legen Sie die benötigten PROFINET-IO Module an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Device** öffnen.
2. Im Kontextmenü von PROFINET-IO Device **Neu** wählen.
3. Für dieses Beispiel die folgenden passenden Module wählen, um vom PROFINET-IO Device **11 Bytes** zu empfangen und **3 Bytes** zu senden.

PROFINET-IO Modul	Steckplatz
Out 2 Byte_1	1
Out 8 Bytes_2	2
Out 1 Byte_3	3
In 2 Byte_4	4
In 1 Bytes_5	5

So nummerieren Sie die PROFINET-IO Device Module:

1. Rechtsklick auf das erste **PROFINET-IO Device Modul** und im Kontextmenü **Eigenschaften** wählen.
2. Tragen Sie **1** im Feld **Steckplatz** ein.
3. Diesen Schritt für weitere **PROFINET-IO Device Module** wiederholen und fortlaufend nummerieren.

i **Nummerieren** Sie die HIMax PROFINET-IO Device Module bei **1** beginnend in aufsteigender Reihenfolge und ohne Lücken.

Konfiguration der PROFINET-IO Device Ausgangsmodule

i Die Summe der Variablen in Byte muss mit der Größe des jeweiligen Moduls in Byte übereinstimmen.

So konfigurieren Sie das Ausgangsmodul [01] Out 2 Bytes_1

1. Im PROFINET-IO Device das Ausgangsmodul **[01] Out 2 Bytes_1** wählen.
2. Rechtsklicken auf **[01] Out 2 Bytes_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller1	UINT	0	PN_Device_Controller1

Tabelle 50: Variablen im Ausgangsmodul [01] Out 2 Bytes_1

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Ausgangsmodul [02] Out 8 Bytes_2

1. Im PROFINET-IO Device das Ausgangsmodul **[02] Out 8 Bytes_2** wählen.
2. Rechtsklicken auf **[02] Out 8 Bytes_2** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller2	DWORD	0	PN_Device_Controller2
PN_Device_Controller3	DWORD	4	PN_Device_Controller3

Tabelle 51: Variablen Ausgangsmodul [02] Out 8 Bytes_2

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Ausgangsmodul [03] Out 1 Byte_3

1. Im PROFINET-IO Device das Ausgangssmodul **[03] Out 1 Byte_3** wählen.
2. Rechtsklicken auf **[03] Out 1 Byte_3** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller4	Byte	0	PN_Device_Controller4

Tabelle 52: Variablen im Ausgangsmodul [03] Out 1 Byte_3

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

Konfiguration der PROFINET-IO Device Eingangsmodule

So konfigurieren Sie das Eingangsmodul [04] In 2 Bytes_4

1. Im PROFINET-IO Device das Eingangsmodul **[04] In 2 Bytes_4** wählen.
2. Rechtsklicken auf **[04] In 2 Bytes_4** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Controller_Device1	UINT	0	PN_Controller_Device1

Tabelle 53: Variablen im Eingangsmodul [04] In 2 Bytes_4

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Eingangsmodul [05] In 1 Byte_5

1. Wählen Sie im PROFINET-IO Device das Eingangsmodul **[05] In 1 Byte_5**.
2. Rechtsklicken auf **[05] In 1 Byte_5** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Controller_Device2	BYTE	0	PN_Controller_Device2

Tabelle 54: Variablen im Eingangsmodul [05] In 1 Byte_5

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

PROFINET-IO Device Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, PROFINET-IO Device** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

i Die Konfiguration des PROFINET-IO Device müssen Sie mit dem Anwenderprogramm der PROFINET-IO Device Ressource neu compilieren und in die Steuerungen laden, damit sie für die PROFINET-IO Kommunikation wirksam werden.

5.9.2 Anlegen des HIMA PROFINET-IO Controller in SILworX

So legen Sie einen neuen HIMA PROFINET-IO Controller an:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFINET-IO Controller** wählen, um einen neue PROFINET-IO Controller hinzuzufügen.
3. Im Kontextmenü vom PROFINET-IO Controller **Eigenschaften** wählen.
4. **COM Modul** auswählen.

Anlegen des HIMA PROFINET-IO Device im Controller

So legen Sie ein HIMax PROFINET-IO Device im PROFINET-IO Controller an:

1. Im Kontextmenü des PROFINET-IO Controller **Neu**, **PROFINET-IO Device** wählen.

So fügen Sie die GSDML-Bibliotheksdatei aus einer externen Datenquelle ein (z.B. CD, USB-Stick, Internet):

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **GSDML-Bibliothek** wählen.
2. Im Kontextmenü der GSDML-Bibliothek**, GSDML Datei hinzufügen** wählen und die zum PROFINET-IO Device zugehörige GSDML-Datei einlesen.

i Die GSDML-Bibliotheksdatei beinhaltet meistens mehrere Geräte eines Herstellers.
Die HIMA GSDML-Bibliotheksdatei finden Sie auf der HIMA Webseite, GSDML-V2.1-Hima-embex-20090805.xml.

So laden Sie die GSDML-Datei für das PROFINET-IO Device:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **PROFINET-IO Device** wählen.
2. Im Kontextmenü **Eigenschaften** wählen und das Register Parameter öffnen.
   - IP-Adresse des PROFINET-IO Device eintragen.
   - Im Dropdown-Menü **GSDML-Datei**, die zum PROFINET-IO Device zugehörige GSDML-Bibliotheksdatei auswählen und **Eigenschaften** schließen.

So wählen Sie den Data Access Point (DAP) für das PROFINET-IO Device:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **PROFINET-IO Device, DAP Modul** wählen.
2. Im Kontextmenü **Device Access Point (DAP) auswählen** wählen und den passenden Datensatz zum PROFINET-IO Device auswählen.

Anlegen der HIMax PROFINET-IO Controller Module

Im PROFINET-IO Controller müssen Sie die Anzahl der tatsächlich zu übertragenden Bytes konfigurieren. Dies erreichen Sie durch Auswahl von *Modulen*, bis die physikalische Konfiguration des Devices erreicht ist.

So legen Sie die benötigten PROFINET-IO Module an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Controller**, **PROFINET-IO Device** öffnen.
2. Im Kontextmenü **Module einfügen** wählen.
3. Für dieses Beispiel die folgenden passenden Module wählen, um vom PROFINET-IO Device **11 Bytes** zu empfangen und **3 Bytes** zu senden.

PROFINET-IO Modul	Steckplatz
DAP Modul (Device access point)	0
Input 2 Byte: Modul_1	1
Input 8 Byte: Modul_2	2
Input 1 Byte: Modul_3	3
Output 2 Byte: Modul_4	4
Output 1 Byte: Modul_5	5

So nummerieren Sie die PROFINET-IO Module:

1. Rechtsklick auf das erste **PROFINET-IO Modul** und im Kontextmenü **Eigenschaften** wählen.
2. Tragen Sie **0** im Feld **Steckplatz** ein.
3. Diesen Schritt für weitere **PROFINET-IO Modul** wiederholen und fortlaufend nummerieren.

Nummerieren Sie die HIMax PROFINET-IO Module bei **0** beginnend in aufsteigender Reihenfolge und ohne Lücken.

Konfiguration der PROFINET-IO Controller Eingangsmodule

Die Summe der Variablen in Byte muss mit der Größe des jeweiligen Moduls in Byte übereinstimmen.

In den Submodulen der Input und Output Module werden die voreingestellten standard Kommunikationsbeziehungen Default Input CR und Default Output CR für dieses Beispiel übernommen.

So konfigurieren Sie das Eingangsmodul [001] Input 2 Byte: Modul_1

1. Im PROFINET-IO Device das Eingangsmodul **[001] Input 2 Byte: Modul_1, [00001] Submodul Inputs_1** wählen.
2. Rechtsklicken auf **[00001] Submodul Inputs_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller1	UINT	0	PN_Device_Controller1

Tabelle 55: Variablen im Eingangsmodul [001] Input 2 Byte: Modul_1

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Eingangsmodul [002] Input 8 Byte: Modul_2

1. Im PROFINET-IO Device das Eingangsmodul **[002] Input 8 Byte: Modul_2, [00001] Submodul Inputs_1** wählen.
2. Rechtsklicken auf **[00001] Submodul Inputs_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller2	DWORD	0	PN_Device_Controller2
PN_Device_Controller3	DWORD	4	PN_Device_Controller3

Tabelle 56: Variablen Eingangsmodul [002] Input 8 Byte: Modul_2

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Eingangsmodul [003] Input 1 Byte: Modul_3

1. Im PROFINET-IO Device das Eingangsmodul **[003] Input 1 Byte: Modul_1, [00001] Submodul Inputs_1** wählen.
2. Rechtsklicken auf **[00001] Submodul Inputs_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Device_Controller4	Byte	0	PN_Device_Controller4

Tabelle 57: Variablen im Eingangsmodul [003] Input 1 Byte: Modul_3

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

Konfiguration der PROFINET-IO Controller Ausgangsmodule

So konfigurieren Sie das Ausgangsmodul [004] Output 2 Bytes: Modul_4

1. Im PROFINET-IO Device das Ausgangsmodul **[004] Output 2 Byte: Modul_4, [00001] Submodul Inputs_1** wählen.
2. Rechtsklicken auf **[00001] Submodul Inputs_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Controller_Device1	UINT	0	PN_Controller_Device1

Tabelle 58: Variablen im Ausgangsmodul [004] Output 2 Byte: Modul_4

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Ausgangsmodul [005] Output 1 Bytes: Modul_5

1. Wählen Sie im PROFINET-IO Device das Ausgangsmodul **[005] Output 1 Bytes: Modul_5, [00001] Submodul Inputs_1**.
2. Rechtsklicken auf **[00001] Submodul Inputs_1** und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** ziehen.

Name	Typ	Offset	Globale Variable
PN_Controller_Device2	BYTE	0	PN_Controller_Device2

Tabelle 59: Variablen im Ausgangsmodul [005] Output 1 Bytes: Modul_5

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

5.9.3 Menüfunktion Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des PROFINET-IO Device öffnet den Dialog **Eigenschaften**.

Element	Beschreibung
Typ	PROFINET-IO Device
Name	Beliebiger, eindeutiger Name, für ein PROFINET-IO Device
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und CPU ausgetauscht werden. Ist die *Refresh Rate* Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 4 ...(2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert
Modul	Auswahl des COM-Moduls auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld *Max. CPU Load [%]* übernehmen. Deaktiviert: Kein Limit der CPU-Last, für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
RPC Port Server	Remote Procedure Call Port Wertebereich: 1024 ... 65535 Standardwert: 49152 RPC Port Server und RPC Port Client dürfen nicht identisch sein!
RPC Port Client	Remote Procedure Call Port Wertebereich: 1024 ... 65535 Standardwert: 49153 RPC Port Server und RPC Port Client dürfen nicht identisch sein!
RT Port Controller	RT Port Wertebereich: 1024 ... 65535 Standardwert: 34962

Tabelle 60: Allgemeine Eigenschaften des PROFINET-IO Device

5.9.4 PROFINET-IO Module

Die folgenden PROFINET-IO Module stehen im HIMA PROFINET-IO Device zur Verfügung.

PROFINET-IO Modul	Max. Größe Eingangsvariablen	Max. Größe Ausgangssvariablen
In 1 Byte	1 Byte	
In 2 Bytes	2 Bytes	
In 4 Bytes	4 Bytes	
In 8 Bytes	8 Bytes	
In 16 Bytes	16 Bytes	
In 32 Bytes	32 Bytes	
In 64 Bytes	64 Bytes	
In 128 Bytes	128 Bytes	
In 256 Bytes	256 Bytes	
In 512 Bytes	512 Bytes	
In 1024 Bytes	1024 Bytes	
In-Out 1 Byte	1 Byte	1 Byte
In-Out 2 Bytes	2 Bytes	2 Bytes
In-Out 4 Bytes	4 Bytes	4 Bytes
In-Out 8 Bytes	8 Bytes	8 Bytes
In-Out 16 Bytes	16 Bytes	16 Bytes
In-Out 32 Bytes	32 Bytes	32 Bytes
In-Out 64 Bytes	64 Bytes	64 Bytes
In-Out 128 Bytes	128 Bytes	128 Bytes
In-Out 256 Bytes	256 Bytes	256 Bytes
In-Out 512 Bytes	512 Bytes	512 Bytes
In-Out 1024 Bytes	1024 Bytes	1024 Bytes
Out 1 Byte		1 Byte
Out 2 Bytes		2 Bytes
Out 4 Bytes		4 Bytes
Out 8 Bytes		8 Bytes
Out 16 Bytes		16 Bytes
Out 32 Bytes		32 Bytes
Out 64 Bytes		64 Bytes
Out 128 Bytes		128 Bytes
Out 256 Bytes		256 Bytes
Out 512 Bytes		512 Bytes
Out 1024 Bytes		1024 Bytes

Tabelle 61: Allgemeine Eigenschaften des PROFINET-IO Device

So legen Sie ein PROFINET-IO Modul an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFINET-IO Device** öffnen.
2. Im Kontextmenü von PROFINET-IO Device **Neu** wählen.
3. Rechtsklick auf PROFINET-IO Modul und **Edit** auswählen.
   - Im Register **Prozessvariablen** die Eingangs- und/oder Ausgangsvariablen einfügen.
   - Im Register **Systemvariablen** können Sie den beiden Systemvariablen globale Variablen zuweisen und diese im Anwenderprogramm verwenden

Element	Typ
Ausgangsdaten gültig	BOOL
Ausgangsdaten vom Controller akzeptiert	BOOL

   - Im Register **Eigenschaften** finden Sie die folgenden Parameter.

Element	Beschreibung
Name	Name des PROFINET-IO Device Moduls
Steckplatz	0 bis 32767
Modul Kennung	Eindeutige Zahl
IO-Typ	1 In 2 Out 3 In-Out
ProcessdataAfterTermination	Wert der Prozessdaten nach einem Verbindungsabbruch - Letzte gültige Prozessdaten behalten - Initialdaten annehmen
Länge der IO-Input Daten	0 ...1024
Länge der IO-Output Daten	0 ...1024

Tabelle 62: Allgemeine Eigenschaften des PROFINET-IO Device

6 PROFIBUS DP

PROFIBUS DP ist ein internationaler, offener Feldbusstandard und wird überall dort eingesetzt, wo eine schnelle Reaktionszeit bei vornehmlich kleinen Datenmengen gefordert wird.

Der HIMA PROFIBUS DP Master und der HIMA PROFIBUS DP Slave erfüllen die Kriterien der europäischen Norm EN 50170 [7] und der weltweit verbindlichen Norm IEC 61158 für PROFIBUS DP.

Der HIMA PROFIBUS DP Master kann zyklisch und azyklisch Daten mit den PROFIBUS DP Slaves austauschen.

Zum azyklischen Datenaustausch stehen Ihnen in SILworX verschiedene Funktionsbausteine zur Verfügung. Mit diesen Funktionsbausteinen können Sie den HIMA PROFIBUS DP Master und die PROFIBUS DP Slaves optimal den Erfordernissen Ihres Projekts anpassen.

Eine redundante PROFIBUS DP Verbindung kann nur durch die Konfiguration eines zweiten PROFIBUS DP Master/Slave und Anpassungen im Anwenderprogramm erreicht werden.

- PROFIBUS DP Master (Siehe Kapitel 6.1)
- PROFIBUS DP Slave (Siehe Kapitel 6.13)

6.1 HIMA PROFIBUS DP Master

Dieses Kapitel beschreibt die Eigenschaften des HIMA PROFIBUS DP Master sowie die Menüfunktionen und Dialoge in SILworX, die Sie zur Konfiguration des HIMA PROFIBUS DP Master benötigen.

Benötigte Ausstattung und Systemanforderungen:

Element	Beschreibung
HIMA Steuerung	HIMax mit COM-Modul
COM-Modul	Das COM-Modul muss an der verwendeten seriellen Feldbusschnittstelle (FB1 oder FB2) mit einem optionalen HIMA PROFIBUS DP Master Submodul ausgerüstet sein, siehe Kapitel 3.7.
Aktivierung	Freischaltung durch Aufsteckmodul, siehe Kapitel 3.5.

Tabelle 63: Benötigte Ausstattung und Systemanforderungen

PROFIBUS DP Master Eigenschaften:

Element	Beschreibung
Typ des HIMA PROFIBUS DP Master	DP-V1 Master Klasse 1 mit zusätzlichen DP-V2 Funktionen
Übertragungsrate	9,6 kbit/s ... 12 Mbit/s
Busadresse	0 ... 125
Max. Anzahl PROFIBUS DP Master	Für jedes COM-Modul können Sie zwei PROFIBUS DP Master konfigurieren.
Max. Anzahl PROFIBUS DP Slaves	Sie können bis zu 122 Slaves pro Ressource (in allen Master Protokollinstanzen) konfigurieren. Hierbei besteht die Restriktion, dass maximal 31 Slaves an ein Bussegment ohne Repeater angeschlossen werden können.
Max. Prozessdatenlänge zu einem Slave	DP-Output =: max. 244 Bytes DP-Input =: max. 244 Bytes

Tabelle 64: Eigenschaften PROFIBUS DP Master

Nach der Norm sind insgesamt drei Repeater zulässig, so dass maximal 122 Busteilnehmer pro serielle Schnittstelle eines Masters möglich sind.

6.1.1 Anlegen eines HIMA PROFIBUS DP Master

So legen Sie einen neuen HIMA PROFIBUS DP Master an:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFIBUS DP Master** wählen, um einen neue PROFIBUS DP Master hinzuzufügen.
3. Im Kontextmenü vom PROFIBUS DP Master **Eigenschaften, Allgemein** auswählen.
4. **Modul** und die **Schnittstelle** auswählen.

6.2 PROFIBUS DP Beispiel

In diesem Beispiel tauscht ein HIMA PROFIBUS DP Master Variablen mit einem HIMA PROFIBUS DP Slave aus.

Hier erfahren Sie, wie Sie den HIMA PROFIBUS DP Master und den HIMA PROFIBUS DP Slave anlegen und parametrieren.

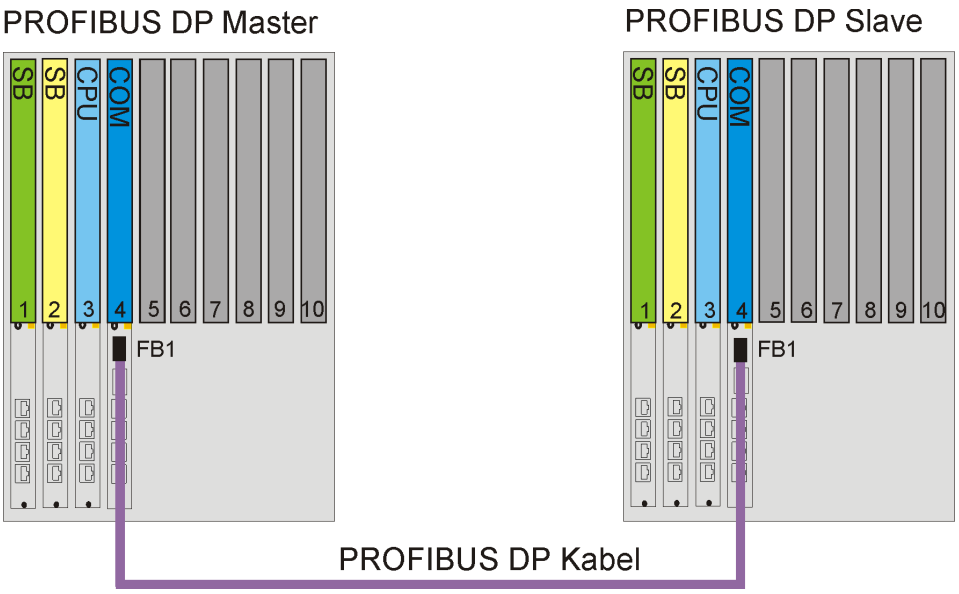


Bild 27: Kommunikation über PROFIBUS DP

Die COM-Module der beiden HIMax Steuerungen müssen an der Feldbusschnittstelle 1 mit dem jeweiligen PROFIBUS DP Submodul ausgestattet sein, siehe Kapitel 3.7.

Folgende Globalen Variablen müssen Sie für dieses Beispiel in SILworX anlegen:

Globale Variable	Typ
PB_Slave_Master1	UINT
PB_Slave_Master2	DWORD
PB_Slave_Master3	DWORD
PB_Slave_Master4	BYTE
PB_Master_Slave1	DWORD
PB_Master_Slave2	BYTE

6.2.1 PROFIBUS DP Slave Konfigurieren

Konfiguration des PROFIBUS DP Slave.

So legen Sie einen neuen HIMA PROFIBUS DP Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFIBUS DP Slave** wählen um einen neuen PROFIBUS DP Slave hinzuzufügen.
3. Im Kontextmenü vom PROFIBUS DP Slave **Edit** wählen.
4. Im Register **Eigenschaften** das **COM-Modul** und die **Schnittstelle** (z. B. FB1) auswählen.

Variablen Zuordnung im HIMA PROFIBUS DP Slave:

1. Im Kontextmenü des PROFIBUS DP Slave **Edit** wählen.
2. Im Dialog **Edit** das Register **Prozessvariablen** wählen.

i Die Startadresse der HIMA PROFIBUS DP Slave Ein- und Ausgangsvariablen beginnt immer bei 0. Erwartet der PROFIBUS DP Master (eines anderen Herstellers) eine höhere Startadresse, müssen Dummy-Variablen vor den Nutzvariablen eingefügt werden.

Ausgänge im HIMA PROFIBUS DP Slave

Name	Typ	Offset	Globale Variable
PB_Slave_Master1	UINT	0	PB_Slave_Master1
PB_Slave_Master2	DWORD	2	PB_Slave_Master2
PB_Slave_Master3	DWORD	6	PB_Slave_Master3
PB_Slave_Master4	BYTE	10	PB_Slave_Master4

Tabelle 65: Ausgänge HIMA PROFIBUS DP Slave

1. In der Objektauswahl die Globalen Variablen zum versenden per Drag&Drop in den Bereich **Ausgangsvariablen** ziehen.

i Die Ausgangsvariablen des HIMA PROFIBUS DP Slave bestehen in diesem Beispiel aus **vier Variablen** mit insgesamt **11 Bytes**. Die Ausgangsvariable mit dem niedrigsten Offset hat die Startadresse **0**.

2. Kontextmenü mit Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen.
3. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

Eingänge im HIMA PROFIBUS DP Slave

Name	Typ	Offset	Globale Variable
PB_Master_Slave1	DWORD	14	PB_Master_Slave1
PB_Master_Slave2	BYTE	18	PB_Master_Slave2

Tabelle 66: Eingänge HIMA PROFIBUS DP Slave

1. In der Objektauswahl die Globalen Variablen zum empfangen per Drag&Drop in den Bereich **Eingangsvariablen** ziehen.

i Die Eingangsvariablen des HIMA PROFIBUS DP Slave bestehen in diesem Beispiel aus **zwei Variablen** mit insgesamt **3 Bytes**. Die Eingangsvariable mit dem niedrigsten Offset hat die Startadresse **0.**

2. Kontextmenü mit Rechtsklick auf eine leere Stelle im Bereich **Eingangsvariablen** öffnen.
3. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

PROFIBUS DP Slave Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, PROFIBUS DP Slave** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

i Die Konfiguration des PROFIBUS DP Slave müssen Sie mit dem Anwenderprogramm der PROFIBUS DP Slave Ressource neu compilieren und in die Steuerungen laden, damit sie für die PROFIBUS DP Kommunikation wirksam werden.

6.2.2 PROFIBUS DP Master Konfigurieren

So legen Sie einen neuen HIMA PROFIBUS DP Master an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFIBUS DP Master** wählen um einen neue PROFIBUS DP Master hinzuzufügen.
3. Im Kontextmenü vom PROFIBUS DP Master **Eigenschaften**, **Allgemein** wählen.
4. Im Register **Allgemein** das **COM-Modul** und die **Schnittstelle** (z. B. FB1) auswählen.

i Die folgenden Schritte dienen der Konfiguration des HIMax PROFIBUS DP Slave im HIMax PROFIBUS DP Master.

So legen Sie einen HIMax PROFIBUS DP Slave im PROFIBUS DP Master an:

1. Im Kontextmenü des PROFIBUS DP Master **Neu**, **PROFIBUS DP Slave** wählen.

So lesen Sie die GSD-Datei für den neuen PROFIBUS DP Slave ein:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master**, **PROFIBUS Slave** öffnen.
2. Im Kontextmenü vom PROFIBUS DP Master **GSD-Datei einlesen** wählen und die zum PROFIBUS Slave zugehörige GSD-Datei aus (z. B. hax100ea.gsd) auswählen.

i Die GSD-Dateien für HIMax Steuerungen finden Sie auf der HIMA Webseite www.hima.com.

Anlegen der HIMax PROFIBUS DP Module

Im PROFIBUS DP Master müssen Sie die Anzahl der tatsächlich zu übertragenden Bytes konfigurieren. Dies erreichen Sie durch Auswahl von *Modulen*, bis die physikalische Konfiguration des Slaves erreicht ist.

i Es ist nicht von Bedeutung, wie viele Module verwendet werden, um auf die erforderliche Anzahl an Bytes zu kommen, so lange die Anzahl von maximal 32 Modulen nicht überschritten wird.

Um die Konfiguration des PROFIBUS DP Master nicht unnötig zu erschweren, sollte die Zahl der gewählten Module möglichst klein gehalten werden.

So legen Sie die benötigten PROFIBUS DP Module an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master**, **PROFIBUS Slave** öffnen.
2. In der Menüleiste **PROFIBUS DP Master**, **Module einfügen** wählen.
3. Für dieses Beispiel die folgenden passenden Module wählen, um vom PROFIBUS DP Slave **11 Bytes** zu empfangen und **3 Bytes** zu senden.

So nummerieren Sie die PROFIBUS DP Module:

1. Rechtsklick auf das erste **PROFIBUS DP Modul** und im Kontextmenü **Eigenschaften** wählen.
2. Tragen Sie **0** im Feld **Steckplatz** ein.
3. Diesen Schritt für weitere **PROFIBUS DP Modul** wiederholen und fortlaufend nummerieren.

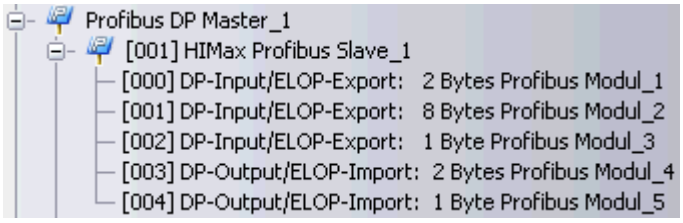

Profibus DP Master_1
[001] HIMax Profibus Slave_1
[000] DP-Input/ELOP-Export: 2 Bytes Profibus Modul_1
[001] DP-Input/ELOP-Export: 8 Bytes Profibus Modul_2
[002] DP-Input/ELOP-Export: 1 Byte Profibus Modul_3
[003] DP-Output/ELOP-Import: 2 Bytes Profibus Modul_4
[004] DP-Output/ELOP-Import: 1 Byte Profibus Modul_5

Bild 28: HIMax PROFIBUS DP Slave mit Modulen

i **Nummerieren** Sie die HIMax PROFIBUS DP Module bei **0** beginnend in aufsteigender Reihenfolge und ohne Lücken.

Die Reihenfolge der PROFIBUS DP Module ist dabei für die Funktion nicht von Bedeutung. Zur besseren Übersicht sollten jedoch die DP-Input-Module und die DP-Output-Module geordnet angelegt werden.

Konfiguration der Eingangs-/Ausgangsmodule

i Die Summe der Variablen in Byte muss mit der Größe des jeweiligen Moduls in Byte übereinstimmen.

So konfigurieren Sie das Eingangsmodul [000] DP-Input/ELOP-Export: 2 Bytes:

1. Im PROFIBUS DP Slave das Eingangsmodul **[000] DP-Input/ELOP-Export: 2 Bytes** wählen
2. Rechtsklicken auf Eingangsmodul und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** des Eingangsmoduls **[000] DP-Input/ELOP-Export: 2 Bytes** ziehen.

Name	Typ	Offset	Globale Variable
PB_Slave_Master1	UINT	0	PB_Slave_Master1

Tabelle 67: Variablen Eingangsmodul [000] DP-Input/ELOP-Export: 2 Bytes

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Eingangsmodul [001] DP-Input/ELOP-Export: 8 Bytes:

1. Im PROFIBUS DP Slave das Eingangsmodul **[001] DP-Input/ELOP-Export: 8 Bytes** wählen.
2. Rechtsklicken auf Eingangsmodul und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** des Eingangsmoduls **[001] DP-Input/ELOP-Export: 8 Bytes** ziehen.

Name	Typ	Offset	Globale Variable
PB_Slave_Master2	DWORD	0	PB_Slave_Master2
PB_Slave_Master3	DWORD	4	PB_Slave_Master3

Tabelle 68: Variablen Eingangsmodul [001] DP-Input/ELOP-Export: 8 Bytes

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Eingangsmodul [002] DP-Input/ELOP-Export: 1 Byte:

1. Im PROFIBUS DP Slave das Eingangsmodul **[002] DP-Input/ELOP-Export: 1 Byte** wählen.
2. Rechtsklicken auf Eingangsmodul und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Eingangssignale** des Eingangsmoduls **[002] DP-Input/ELOP-Export: 1 Byte** ziehen.

Name	Typ	Offset	Globale Variable
PB_Slave_Master4	BYTE	0	PB_Slave_Master4

Tabelle 69: Variablen Eingangsmodul [002] DP-Input/ELOP-Export: 1 Byte

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Ausgangsmodul [003] DP-Output/ELOP-Import 2 Bytes:

1. Wählen Sie im PROFIBUS DP Slave das Ausgangsmodul **[003] DP-Output/ELOP-Import 2 Bytes**.
2. Rechtsklicken auf Ausgangsmodul und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** des Ausgangsmoduls **[003] DP-Output/ELOP-Import 2 Bytes** ziehen.

Name	Typ	Offset	Globale Variable
PB_Master_Slave1	UINT	0	PB_Master_Slave1

Tabelle 70: Variablen Ausgangsmodul [003] DP-Output/ELOP-Import: 2 Bytes

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

So konfigurieren Sie das Ausgangsmodul [004] DP-Output/ELOP-Import 1 Byte:

1. Wählen Sie im PROFIBUS DP Slave das Ausgangsmodul **[004] DP-Output/ELOP-Import 1 Byte**.
2. Rechtsklicken auf Ausgangsmodul und im Kontextmenü **Edit** wählen.
3. Im Dialog **Edit** das Register **Prozessvariablen** wählen.
4. In der Objektauswahl die passende Variable wählen und per Drag&Drop in den Bereich **Ausgangssignale** des Ausgangsmoduls **[004] DP-Output/ELOP-Import 1 Byte** ziehen.

Name	Typ	Offset	Globale Variable
PB_Master_Slave2	BYTE	0	PB_Master_Slave2

Tabelle 71: Variablen Ausgangsmodul [004] DP-Output/ELOP-Import: 1 Byte

5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen.
6. Im Kontextmenü **Neue Offsets** wählen, damit die Offsets der Variablen neu generiert werden.

Anlegen der Benutzerdaten im PROFIBUS DP Master

So legen Sie die Benutzerdaten im PROFIBUS DP Master an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master** öffnen.
2. Rechtsklick auf **PROFIBUS Master** und im Kontextmenü **Eigenschaften** wählen.
3. Register **Daten** wählen und die **Schaltfläche ...** neben den Benutzerdaten anklicken.

In dem 32 Byte langen Benutzerdatenfeld werden die *Startadresse* und die *Anzahl der Variablen der Blöcke* definiert, siehe auch Kapitel 6.8.

4. Für dieses Beispiel müssen die folgenden Benutzerdaten eingetragen werden:
   4, damit **Vier Variablen** vom PROFIBUS DP Master empfangen werden.
   2 damit **Zwei Variablen** vom PROFIBUS DP Master gesendet werden.
   Die Startadresse des Eingangs- und Ausgangsblocks beginnt jeweils bei **0**.

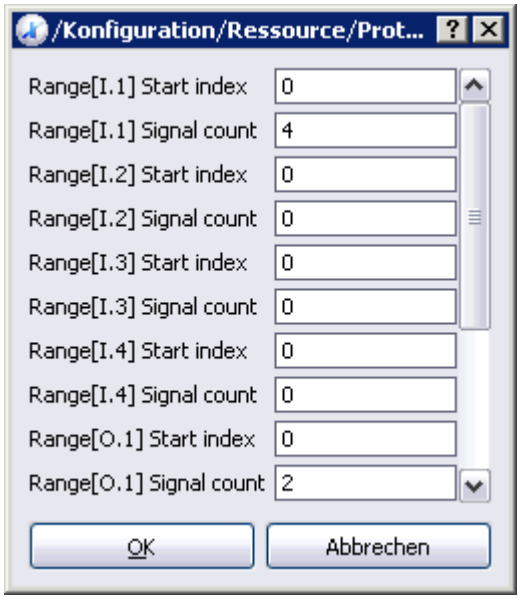


Bild 29: Benutzerdatenfeld

PROFIBUS DP Slave Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, PROFIBUS DP Master** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

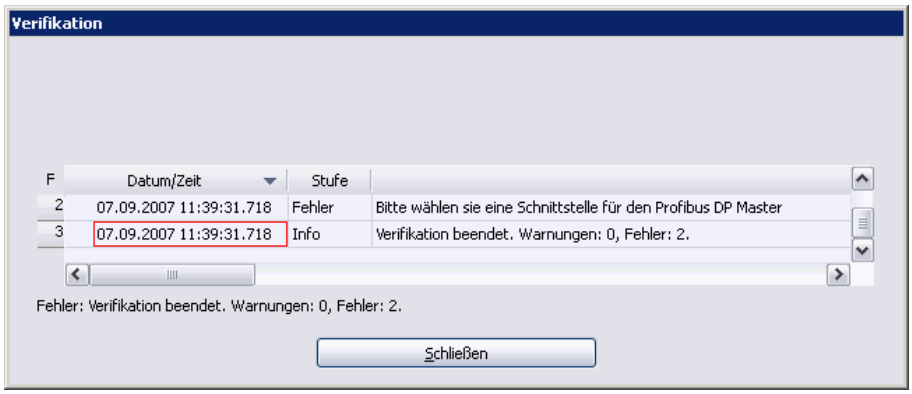

Verifikation
F
Datum/Zeit
Stufe
2 07.09.2007 11:39:31.718 Fehler Bitte wählen sie eine Schnittstelle für den Profibus DP Master
3 07.09.2007 11:39:31.718 Info Verifikation beendet. Warnungen: 0, Fehler: 2.
Fehler: Verifikation beendet. Warnungen: 0, Fehler: 2.
Schließen

Bild 30: Dialogfenster Verifikation

i Die Konfiguration des PROFIBUS DP Master müssen Sie mit dem Anwenderprogramm der PROFIBUS DP Master Ressource neu compilieren und in die Steuerungen laden, damit sie für die PROFIBUS DP Kommunikation wirksam werden.

Optimieren der PROFIBUS DP Parameter

Mit den Standardwerten der PROFIBUS Parameter ist eine PROFIBUS Kommunikation vermutlich problemlos möglich. Durch eine weitere Optimierung der Einstellungen können Sie jedoch einen schnelleren Datenaustausch und eine bessere Fehlererkennung erreichen.

So ermitteln Sie die tatsächliche Target Rotation Time TTR [ms]

1. Control Panel der HIMax PROFIBUS DP Master öffnen.
2. Im Strukturbaum des Control Panels **PROFIBUS DP Master** wählen und die tatsächliche **Target Rotation Time TTR [ms]** ablesen. Notieren Sie sich diesen Wert.

So ermitteln Sie benötigte Parameter vom PROFIBUS DP Slave

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master, PROFIBUS DP Slave** öffnen.
2. Rechtsklick auf **HIMax PROFIBUS Slave** und wählen Sie **Eigenschaften**.
3. Register **Features** wählen und Parameter **Min. Slave Interval MSI [ms]** für diesen PROFIBUS DP Slave ablesen. Notieren Sie sich diesen Wert.
4. Register **Übertragungsraten** wählen und Parameter **Max. Tsdr** für die verwendete Übertragungsrate ablesen. Notieren Sie sich diesen Wert.

So tragen Sie die ermittelten Parameter ein

1. Rechtsklick auf **PROFIBUS Master** und im Kontextmenü **Eigenschaften**.
2. Register **Zeiten** wählen.

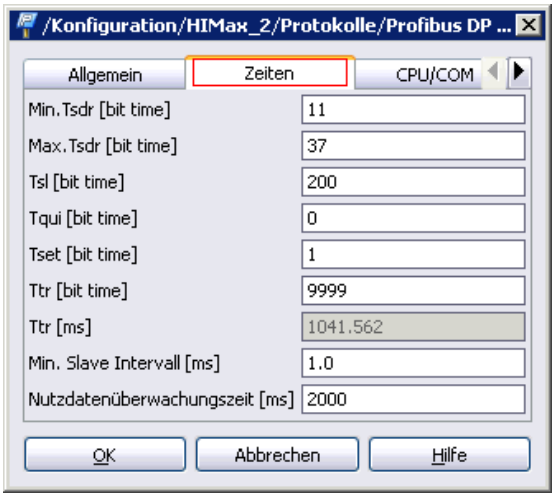


Bild 31: Eigenschaften des PROFIBUS DP Master

3. Abgelesene **Max. Tsdr** in **bit Time** umrechnen und eintragen.
4. Abgelesene **Target Rotation Time TTR [ms]** in **bit Time** umrechnen und mit zuzüglich 1/3 Sicherheitszuschlag in das Feld **Target Rotation Time TTR [ms]** eintragen.
5. Abgelesene **Min. Slave Interval MSI [ms]** eintragen.

i Sind mehrere Slaves konfiguriert, muss der höchste Wert von MaxTsdr [bit time] und der größte von Min. Slave Intervall [ms] verwendet werden.

6. Die Nutzdatenüberwachungszeit [ms] ist ≥ 6*Ttr [ms] einzustellen.

Watchdog-Zeit für PROFIBUS-DP Slave eintragen

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master, PROFIBUS DP Slave** öffnen.
2. Rechtsklick auf **HIMax PROFIBUS Slave** und wählen Sie **Eigenschaften**.

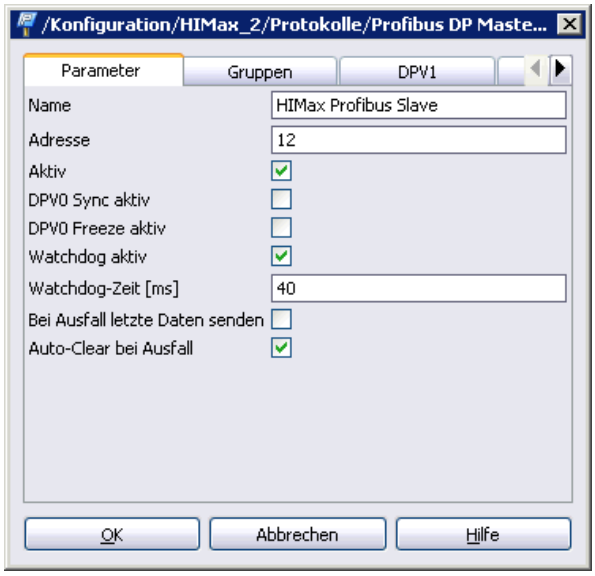


Bild 32: Eigenschaften des PROFIBUS DP Slave

3. Register **Parameter** wählen und im Kontrollkästchen **Watchdog aktiv** Häkchen setzen.
4. Watchdog-Zeit [ms] ≥ 6*Ttr [ms] im Feld **Watchdog-Zeit [ms]** eintragen.

i Die Konfiguration des PROFIBUS DP Master und Slave müssen Sie mit dem Anwenderprogramm der PROFIBUS DP Master und Slave Ressourcen neu compilieren und in die Steuerungen laden, damit sie für die PROFIBUS DP Kommunikation wirksam werden.

6.3 Menüfunktionen des PROFIBUS DP Master

6.3.1 Edit

Die Menüfunktion **Edit** aus dem Kontextmenü des PROFIBUS DP Master öffnet den Dialog **Edit**.

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFIBUS DP Master im Anwenderprogramm auszuwerten.

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td>Anzahl Fehler</td><td>Anzahl Fehler seit Reset der Statistik.</td></tr>
<tr><td>Baud-Rate</td><td>Baudrate (bit/s), mit welcher der Bus betrieben wird.</td></tr>
<tr><td>Bus Fehler</td><td>Tritt ein Busfehler auf, wird in der Systemvariable Bus Fehler ein Fehlercode gesetzt. Ein Fehlercode bleibt solange anliegen, bis der Busfehler behoben ist.
<table>
<tr><th>Code</th><th>Bedeutung</th></tr>
<tr><td>0</td><td>OK, kein Busfehler</td></tr>
<tr><td>1</td><td>Adressfehler:
Adresse des Masters auf dem Bus bereits vorhanden</td></tr>
<tr><td>2</td><td>Busstörung
Störung auf dem Bus registriert, (z. B. Bus nicht richtig abgeschlossen, mehrere Teilnehmer senden gleichzeitig).</td></tr>
<tr><td>3</td><td>Protokollfehler
fehlerhaft codiertes Paket empfangen.</td></tr>
<tr><td>4</td><td>Hardwarefehler
Hardware hat Fehler gemeldet, z. B. bei zu kurz eingestellten Zeiten</td></tr>
<tr><td>5</td><td>Unbekannter Fehler
Master hat Zustand aus unbekanntem Grund gewechselt</td></tr>
<tr><td>6</td><td>Controller Reset
Controller-Chip wird bei schweren Busstörungen zurückgesetzt.</td></tr>
</table>
Um die Statusvariable Bus Fehler im Anwenderprogramm auszuwerten muss diese mit einer Variablen verbunden werden.</td></tr>
<tr><td>Durchschnittliche Zykluszeit</td><td>Gemessene durchschnittliche Buszykluszeit in Millisekunden.</td></tr>
<tr><td>Letzte Zykluszeit</td><td>Gemessene Buszykluszeit in Millisekunden.</td></tr>
<tr><td>Master-Status</td><td>zeigt den momentanen Protokollzustand an.
0: OFFLINE
1: STOP
2: CLEAR
3: OPERATE
Um die Statusvariable Master-Status im Anwenderprogramm auszuwerten muss dieses mit einer Variablen verbunden werden.</td></tr>
<tr><td>Maximale Zykluszeit</td><td>Gemessene maximale Buszykluszeit in Millisekunden.</td></tr>
<tr><td>Min. Slave-Intervall</td><td>Gemessenes minimales Slave-Intervall eines diesem Master zugeordneten Slaves.</td></tr>
<tr><td>Minimale Zykluszeit</td><td>Gemessene minimale Buszykluszeit in Millisekunden.</td></tr>
<tr><td>Target rotation time</td><td>Projektierte Token Umlaufzeit.</td></tr>
</table>

Tabelle 72: Systemvariablen des PROFIBUS DP Master

6.3.2 Menüfunktion Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des PROFIBUS DP Master öffnet den Dialog **Eigenschaften**.

Das Dialogfenster enthält die folgenden Register:

Register Allgemein

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td>Typ</td><td>PROFIBUS DP Master</td></tr>
<tr><td>Name</td><td>Beliebiger, eindeutiger Name, für einen PROFIBUS DP Master</td></tr>
<tr><td>Modul</td><td>Auswahl des COM-Moduls auf dem dieses Protokoll abgearbeitet wird.</td></tr>
<tr><td>Max. CPU Load verwenden</td><td>Aktiviert:
Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen.
Deaktiviert:
Kein Limit der CPU-Last, für dieses Protokoll verwenden.</td></tr>
<tr><td>Max. CPU Load [%]</td><td>Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf.

Wertebereich: 1 ... 100%
Standardwert: 30%</td></tr>
<tr><td>Adresse</td><td>Stationsadresse des Masters.
Die Stationsadresse des Masters darf auf dem Bus nur einmal vorhanden sein.

Wertebereich: 0 ... 125
Standardwert: 0</td></tr>
<tr><td>Schnittstelle</td><td>COM-Schnittstelle, die für den Master benutzt werden soll.
Wertebereich: FB1, FB2</td></tr>
<tr><td>Baudrate</td><td>Baudrate (Bit/s), mit welcher der Bus betrieben wird.
Mögliche Werte:
<table>
<tr><th>Wert</th><th>Baudrate</th><th>FB1</th><th>FB2</th></tr>
<tr><td>9600</td><td>9,6 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>19200</td><td>19,2 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>45450</td><td>45,45 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>93750</td><td>93,75 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>187500</td><td>187,5 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>500000</td><td>500 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>1500000</td><td>1,5 Mbit/s</td><td>X</td><td>X</td></tr>
<tr><td>3000000</td><td>3 Mbit/s</td><td>X</td><td>-</td></tr>
<tr><td>6000000</td><td>6 Mbit/s</td><td>X</td><td>-</td></tr>
<tr><td>12000000</td><td>12 Mbit/s</td><td>X</td><td>-</td></tr>
</table></td></tr>
</table>

Tabelle 73: Allgemeine Eigenschaften des PROFIBUS DP Master

Register Zeiten

Element	Beschreibung
MinTsdr [bit time]	Min. Station Delay Time: Zeit, die ein PROFIBUS DP Slave mindestens warten muss, bevor er antworten darf. Wertebereich: 11 ... 1023 Standardwert: 11
MaxTsdr [bit time]	Max. Station Delay Time: Zeit, die ein PROFIBUS DP Slave maximal benötigen darf, um zu antworten. Max Tsdr ≥ Tsdr (des angeschlossenen Slaves mit der höchsten Tsdr) Die MaxTsdr-Werte der Slaves werden aus den GSD-Dateien gelesen und befinden sich im Dialog **Eigenschaften** der Slaves im Register **Übertragungsraten**. Wertebereich: 37 ... 65525 Standardwert: 37
Tsl [bit time]	Slot Time Maximale Zeitspanne, in welcher der Master auf eine Antwort des Slaves wartet. Tsl > MaxTsdr + 2*Tset +Tqui + 13 Wertebereich: 37 ... 16383 Standardwert: 37
Tqui [bit time]	Quiet Time for Modulator (Modulatorausklingzeit) Zeit, die ein Teilnehmer für das Umschalten von Senden auf Empfangen benötigen darf. Wertebereich: 0 ... 493 Standardwert: 0
Tset [bit time]	Setup Time Reaktionszeit auf ein Ereignis Wertebereich: 1 ... 494 Standardwert: 1
Ttr [bit time]	Projektierte Zeit für einen Token-Umlauf Die maximale zur Verfügung gestellte Zeit für einen Token-Umlauf. Eine untere Abschätzung der Ttr erhält man durch eine Berechnung, siehe Kapitel 6.4.4. Wertebereich: 256 ... 16777215 Standardwert: 999
Ttr [ms]	Tatsächliche Token-Umlaufzeit in ms
Min. Slave Intervall [ms]	Mindestzeit, die zwischen zwei zyklischen Abfragen eines Slaves verstreichen muss. Diese wird vom Master eingehalten und auf keinen Fall unterschritten. Der PROFIBUS DP Zyklus kann sich jedoch verlängern, wenn der Isochron Mode inaktiv ist und der Anteil der azyklischen Telegramme in einem Zyklus ansteigt. Der Wert für *Min. Slave Intervall* des Slaves wird aus der GSD-Datei gelesen und befindet sich im Dialog **Eigenschaften** des Slaves im Register **Features**. Im Isochron Mode gibt *Min. Slave Intervall* die Zeit des isochronen Zyklus vor.
Element	Beschreibung
Min. Slave Intervall [ms] (Fortsetzung)	Der Isochron Mode wird aktiviert wenn Isochron Mode Sync oder Freeze aktiviert sind. Siehe auch Aktualisierungszeit zwischen CPU und COM (Register CPU/COM). Wertebereich: 0 ... 6553.5 Standardwert: 1.0

Nutzdaten-überwachungszeit [ms]	Zeitspanne, innerhalb welcher der Master seinen momentanen Zustand auf dem Bus mitteilen muss. Richtwert: Nutzdatenüberwachungszeit = WDZ des Slaves Wertebereich: 0 ... 65535 [10ms] Standardwert: 2000

Tabelle 74: Register Zeiten im Eigenschaftendialog des PROFIBUS DP Master

Register CPU/COM

Die Vorgabewerte für die Parameter sorgen für den schnellstmöglichen Datenaustausch der PROFIBUS DP Daten zwischen dem COM-Modul (COM) und dem CPU-Modul (CPU) in der HIMax Steuerung. Diese Parameter sollten nur dann geändert werden, wenn eine Reduzierung der COM- und/oder CPU-Auslastung für eine Anwendung erforderlich ist und der Prozess dies zulässt.

i Die Änderung der Parameter wird nur für den erfahrenen Programmierer empfohlen. Eine Erhöhung der COM und CPU Aktualisierungszeit bedeutet auch, dass die tatsächliche Aktualisierungszeit der PROFIBUS DP Daten erhöht wird. Zeitanforderungen der Anlage prüfen.

Beachten Sie auch den Parameter *Min. Slave Intervall [ms]*, der die Aktualisierungszeit der PROFIBUS DP Daten vom/zum PROFIBUS DP Slave festlegt. Dieser kann entsprechend der COM/CPU-Aktualisierungszeit erhöht werden.

Element	Beschreibung
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und CPU ausgetauscht werden. Ist die *Refresh Rate* Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 0 ... (2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert

Tabelle 75: Register CPU/COM im Eigenschaftendialog des PROFIBUS DP Master

Register Sonstige

Element	Beschreibung
Max. Anz. Sendewdh.	Maximale Anzahl an Sendewiederholungen eines Masters, wenn ein Slave nicht antwortet. Wertebereich: 0 ... 7 Standardwert: 1

Highest Active Address	Highest Station Address (HSA) Höchste zu erwartende Stationsadresse eines Masters. Master, mit Stationsadressen jenseits von HSA, werden nicht in den Token-Ring aufgenommen. Wertebereich: 0 ... 125 Standardwert: 125
Isochron Mode Sync	Der Isochron Mode Sync ermöglicht eine taktsynchrone Regelung in Master und Slave und ein zeitgleiches Aktivieren der physikalischen Ausgänge mehrerer Slaves. Ist der Isochron Mode Sync aktiv, dann sendet der Master den Steuerbefehl „Sync“ als Broadcast-Telegramm an alle Slaves. Sobald Slaves, die den Isochron Mode unterstützen, den Steuerbefehl „Sync“ erhalten, schalten sie die Daten aus dem Anwenderprogramm zeitgleich auf die physikalischen Ausgänge. Die Werte der physikalischen Ausgänge bleiben bis zum nächsten Sync-Befehl eingefroren. Die Zykluszeit wird durch das „Min. Slave Intervall“ vorgegeben. Bedingung: Ttr < Min. Slave Intervall Standardwert: Deaktiviert
Isochron Mode Freeze	Der Isochron Mode Freeze ermöglicht eine zeitgleiche Übernahme der Eingangsdaten mehrerer Slaves. Ist der Isochron Mode Freeze aktiv, sendet der Master den Steuerbefehl „Freeze“ als Broadcast-Telegramm an alle Slaves. Sobald die Slaves, die den Isochron Mode unterstützen, den Steuerbefehl „Freeze“ erhalten, werden die Variablen der physikalischen Eingänge auf dem momentanen Wert eingefroren. Die Werte können dann vom Master gelesen werden. Erst nach einem weiteren Steuerbefehl „Freeze“ werden die Eingangsdaten aktualisiert. Die Zykluszeit wird durch das „Min. Slave Intervall“ vorgegeben. Bedingung: Ttr < Min. Slave Intervall Standardwert: Deaktiviert
Auto-Clear bei Fehler	Der Master geht in den Zustand CLEAR, wenn ein Slave ausfällt bei dem Auto-Clear bei Ausfall gesetzt ist. Standardwert: Deaktiviert
Zeitmaster	Der Master ist auch Zeitmaster und versendet die Systemzeit periodisch über den Bus. Standardwert: Deaktiviert
Clock Sync Interval [ms]	Uhr-Synchronistations-Intervall. Zeitspanne, innerhalb welcher der Zeitmaster die Systemzeit auf dem Bus versendet. Wertebereich: 0 ... 65535 Standardwert: 0

Tabelle 76: Sonstige Eigenschaften des PROFIBUS DP Master

6.4 Die PROFIBUS-DP Buszugriffsverfahren

Das Buszugriffsverfahren stellt jedem Busteilnehmer ein definiertes Zeitfenster zur Verfügung, in dem der Busteilnehmer seine Kommunikationsaufgabe erfüllen muss.

6.4.1 Master/Slave-Protokoll

Die Buszuteilung zwischen einem PROFIBUS-DP Master und einem PROFIBUS-DP Slave wird über das Master/Slave-Verfahren sichergestellt.

Ein aktiver PROFIBUS-DP Master kommuniziert mit passiven PROFIBUS-DP Slaves.

Der PROFIBUS-DP Master, der das Token besitzt, hat die Sendeberechtigung und kann mit dem ihm zugewiesenen PROFIBUS-DP Slaves kommunizieren. Der Master teilt einem Slave den Bus für bestimmte Zeit zu, innerhalb welcher der Slave antworten muss.

6.4.2 Token-Protokoll

Die Buszuteilung zwischen Automatisierungsgeräten (Master Klasse 1) und/oder Programmiergeräten (Master Klasse 2) wird über Token Passing sichergestellt.

Alle PROFIBUS-DP Master, die gemeinsam an einem Bus angeschlossen sind, bilden einen Token-Ring. Der aktive PROFIBUS-DP Master, der im Besitz des Tokens ist, übernimmt in dieser Zeit die Masterfunktion am Bus.

Die PROFIBUS-DP Master werden im Token-Ring nach aufsteigenden Stationsadressen geordnet und das Token wird in dieser Reihenfolge bis zum PROFIBUS-DP Master mit der höchsten Stationsadresse weitergegeben.

Dieser gibt das Token an den Master mit der niedrigsten Stationsadresse weiter, um den Token-Ring zu schließen.

Die Token-Umlaufzeit entspricht dem einmaligen Umlauf des Token über alle PROFIBUS-DP Master. Die Target Rotation Time Ttr ist die maximal erlaubte Zeit für einen Token-Umlauf.

6.4.3 Token-Umlaufzeit (Ttr)

Richtwerte für verschiedene Übertragungsraten.

Bei der Konfiguration des PROFIBUS DP Master beachten, dass ein Teil der Parameter im Register **Zeiten** von der im Register **Allgemein** eingestellten Baudrate abhängt. Verwenden Sie für die erste (initial) Konfiguration die in der folgenden Tabelle angegebenen Richtwerte. In einem späteren Schritt werden die Werte optimiert.

	9,6k	19,2k	45,45k	93,75k	187,5k	500k	1,5M	3M	6M	12M
MinTsdr	11	11	11	11	11	11	11	11	11	11
MaxTsdr	60	60	400	60	60	100	150	250	450	800
Tsl bit time	100	100	640	100	100	200	300	400	600	1000
Tqui bit time	0	0	0	0	0	0	0	3	6	9
Tset bit time	1	1	95	1	1	1	1	4	8	16

Tabelle 77: HIMax Richtwerte für Token-Umlaufzeit bei verschiedenen Übertragungsraten

Alle Zeitangaben in sind in Tbit angegeben (1Tbit = 1/[bit/s]).

MinTsdr ist mindestens 11 Tbit lang, da ein Zeichen aus 11 Bits (1 Startbit, 1 Stoppbit, 1 Paritätsbit, 8 Datenbits) besteht.

Übertragungszeit für ein Zeichen

Baudrate	Tbit bit = 1/Baudrate	Zeit
9600 bit/s	1 / 9600 = 104,166 µs	11*104,166 µs = 114,583 ms
6 Mbit/s	1/ 6*1066 = 166,667 ns	11*166,667 ns = 1,833 µs

Tabelle 78: Übertragungszeit für ein Zeichen bei verschiedenen Übertragungsraten

6.4.4 Token-Umlaufzeit Ttr berechnen

Minimale Token-Umlaufzeit Ttr wie folgt berechnen:

Ttr_{min} = n * (198 + T1 + T2) + b * 11 + 242 + T1 + T2 + Tsl

Element	Bedeutung
n	Anzahl aktiver Slaves
b	Anzahl E/A-Datenbytes der aktiven Slaves (Input plus Output)
T0	35 + 2 * Tset + Tqui
T1	Wenn T0 < MinTsdr: T1 = MinTsdr Wenn T0 > MinTsdr: T1 = T0
T2	Wenn T0 < MaxTsdr: T2 = MaxTsdr Wenn T0 > MaxTsdr: T2 = T0
Tsl	Slot Time Maximale Zeitspanne, in welcher der Master auf eine Antwort des Slaves wartet
198	Zwei mal Telegrammkopf des Telegramms mit variabler Länge (für Request und Response)
242	Global_Control, FDL_Status_Req und Token-Weitergabe

Tabelle 79: Elemente zur Berechnung der Token-Umlaufzeit

i Diese Abschätzung der Token-Umlaufzeit *Ttr* gilt nur, wenn nur ein Master am Bus betrieben wird, keine Sendungen wiederholt werden müssen und keine azyklischen Daten übertragen werden.

Stellen Sie auf keinen Fall eine kleinere *Ttr* ein als mit obiger Formel berechnet, da sonst eine fehlerfreie Funktion nicht mehr garantiert werden kann. HIMA empfiehlt, das Doppelte oder Dreifache des berechneten Wertes einzustellen.

Beispiel zur Berechnung der Token-Umlaufzeit *Ttr*

Folgende Konfiguration ist gegeben:

5 aktive Slaves

(n = 5)

20 E/A-Datenbytes pro Slave

(b = 100)

Die folgenden Zeitkonstanten für eine Übertragungsrate von 6 Mbit/s wurden aus der Tabelle 79 entnommen:

- MinTsdr = 11 T_{bit}
- MaxTsdr = 450 T_{bit}
- Tsl bit time = 600 T_{bit}
- Tqui bit time = 6 T_{bit}
- Tset bit time = 8 T_{bit}

T0 = 35 + 2 * Tset + Tqui

T0 = 35 + 2 * 8 + 6

T0 = 57 T_{bit}

Da T0>MinTsdr: **T1 = T0 = 57 T_{bit}**

Da T0<MaxTsdr: **T2 = MaxTsdr = 450 T_{bit}**

Die ermittelten Werte in die Formel für die minimale Token-Umlaufzeit einsetzen:

Ttr_{min} = n * (198 + T1 + T2) + b * 11 + 242 + T1 + T2 + Tsl

Ttr_{min}= 5 (198+57+450)+100*11+242+57+450+600

Ttr_{min} [T_{bit}] = 5974 T_{bit}

Ergebnis:

Ttr_{min} [µs] = 5974 T_{bit} * 166,67 ns = 995,68 µs

i

Ttr wird bei der Eingabe im Dialogfenster geprüft.

Ist der eingetragene Wert *Ttr* kleiner als der von SILworX errechnete Wert, erfolgt eine Fehlermeldung in der Statusanzeige. Zusätzlich wird ein Mindestwert für *Ttr* vorgeschlagen.

Sind *Isochron Mode Sync* oder *Isochron Mode Freeze* ausgewählt, wird die Zykluszeit vom Parameter *MinSlaveInterval* vorgegeben. Die *Ttr* muss dann auf jeden Fall kleiner als das *Min. Slave Intervall* sein.

Das Nichteinhalten dieser Bedingung im Iso-Mode führt zu einer Fehlermeldung.

6.5 Isochroner PROFIBUS DP Zyklus (ab DP-V2)

Der PROFIBUS DP Zyklus besteht hier aus einem festen, zyklischen und einem ereignisbedingten, azyklischen Telegrammteil.

Der azyklische Telegrammteil in einem PROFIBUS DP Zyklus kann diesen entsprechend verlängern, was in bestimmten Anwendungen, wie z. B. in der Antriebstechnik, unerwünscht ist.

Um eine konstante Zykluszeit (t_{const}) zu erreichen, wird im Master der Isochron-Mode aktiviert, bei dem der Parameter *Min. Slave Intervall [ms]* die konstante Zykluszeit (t_{const}) vorgibt. Der so parametrierte isochrone PROFIBUS DP Zyklus besitzt eine Taktgenauigkeit mit einer Abweichung von < 10 ms.

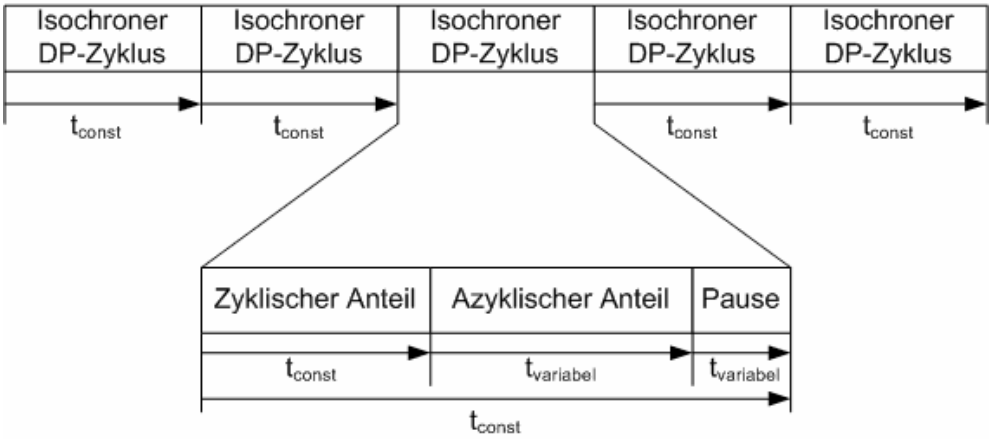


Bild 33: Isochroner PROFIBUS DP Zyklus

Um den zyklischen Anteil zu ermitteln, muss der Anwender die minimale Token-Umlaufzeit berechnen.

Zusätzlich muss ein ausreichend großes Zeitintervall (typischerweise zwei- bis dreimal minimale Token-Umlaufzeit *Ttr*) für den azyklischen Anteil reserviert werden. Wird die reservierte Zeit nicht benötigt, wird eine Pause vor dem nächsten Zyklus eingelegt, um die Zykluszeit konstant zu halten. (siehe auch Kapitel 6.4.3, Token-Umlaufzeit *Ttr*).

i Der Master wird über *Min. Slave Intervall [ms]* mit der vom Anwender ermittelten DP-Zykluszeit parametriert.

Damit der *Isochron Mode* wirksam ist, muss mindestens einer der beiden Parameter *Isochron Mode Sync* oder *Isochron Mode Freeze* im Master aktiviert werden.

An dem Bus darf dann nur ein Master im isochronen Mode betrieben werden. Weitere Master sind nicht zulässig.

6.5.1 Isochron Mode (ab DP-V2)

Diese Funktion ermöglicht eine taktsynchrone Regelung in Master und Slave, unabhängig von der Belastung des Busses. Der Buszyklus wird mit einer Taktabweichungen von < 10 ms synchronisiert. Damit können hochgenaue Positionierungsvorgänge realisiert werden.

i Die Vorteile des *Isochron-Mode* können auch eingeschränkt von Slaves (DP-V0-Slaves) genutzt werden, die den *Isochron-Mode* nicht unterstützen. Man aktiviert dazu bei den *Slaves Sync* und/oder *Freeze* und ordnet sie der Gruppe 8 zu.

Typischerweise verwendet man den *Sync*- und *Freeze* -Mode gleichzeitig.

6.5.2 Isochron Mode Sync (ab DP-V2)

Der *Isochron Mode Sync* ermöglicht eine taktsynchrone Regelung in Master und Slave, und ein zeitgleiches aktivieren der Ausgänge mehrerer Slaves.

6.5.3 Isochrone Mode Freeze (ab DP-V2)

Der *Isochron Mode Freeze* ermöglicht eine zeitgleiche Übernahme der Eingangsdaten mehrerer Slaves.

6.6 Menüfunktionen PROFIBUS DP Slave (im Master)

6.6.1 Anlegen eines PROFIBUS DP Slave (im Master)

So legen Sie im HIMA PROFIBUS DP Master einen PROFIBUS DP Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle, Profibus DP Master** öffnen.
2. Im Kontextmenü von Profibus DP Master **Neu**, **PROFIBUS Slave** wählen, um einen neuen PROFIBUS Slave hinzuzufügen.

6.6.2 Edit

Die Menüfunktion **Edit** aus dem Kontextmenü des PROFIBUS DP Slave öffnet den Dialog **Systemvariablen**.

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFIBUS DP Slave im Anwenderprogramm auszuwerten.

<table>
<tr><th>Element</th><th>Beschreibung</th></tr>
<tr><td>Activation control</td><td>Wechsel von 0 nach 1 deaktiviert den Slave.
Wechsel von 1 nach 0 aktiviert einen zuvor deaktivierten Slave.
Aktiviert = 0
Deaktiviert = 1</td></tr>
<tr><td>PNO Ident Nummer</td><td>Von der PROFIBUS DP Nutzerorganisation e.V. zugeteilte 16-Bit-Nummer, die ein Produkt (Feldgerät) eindeutig kennzeichnet.</td></tr>
<tr><td>Standard Diagnose</td><td>Über die Standard Diagnose teilt der Slave dem Master seinen aktuellen Zustand mit. Diese Variable enthält immer die zuletzt empfangene Standard Diagnose. Die Parameter entsprechen dem Diagnosetelegramm gemäß IEC 61158.</td></tr>
<tr><td>Verbindungszähler</td><td>Wird mit jeder neuen Verbindung inkrementiert.
Zählt ab Reset des Zählers</td></tr>
<tr><td>Verbindungszustand</td><td>
<table>
<tr><th>Wert</th><th>Beschreibung</th></tr>
<tr><td>0</td><td>Deaktiviert:
die Parametersätze für diese Slaves werden geladen, die Slaves jedoch vollkommen ignoriert. Die Inputdaten werden auf die Initialwerte gesetzt, auf dem Bus ist keinerlei Aktivität bzgl. dieser Slaves sichtbar.</td></tr>
<tr><td>1</td><td>Inaktiv (nicht verbunden):
Ist ein Slave nicht (mehr) erreichbar, werden die Inputdaten auf die Initialwerte gesetzt.
Für jeden Slave können die folgenden Optionen gewählt werden:
▪ Master sendet weiterhin Outputdaten zum Slave oder
▪ Master versucht, den Slave neu zu parametrieren.</td></tr>
<tr><td>2</td><td>Aktiv (verbunden):
Slaves tauschen E/A-Daten mit der CPU aus</td></tr>
</table>
</td></tr>
<tr><td>Zähler Slave-Alarme</td><td>Anzahl bisher gelieferter Alarme.
Zählt ab Reset des Zählers.</td></tr>
<tr><td>Zähler Standard Diagnose</td><td>Anzahl bisher gelieferter Diagnosemeldungen.
Zählt ab Reset des Zählers.</td></tr>
</table>

Tabelle 80: Systemvariablen des PROFIBUS DP Slave

6.6.3 Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des PROFIBUS DP Slave öffnet den Dialog **Eigenschaften**. Das Dialogfenster enthält die folgenden Register:

Register Parameter

Element	Beschreibung
Name	Name des Slave
Adresse	Adresse des Slave Wertebereich: 0 ... 125 Standardwert: 0
Aktiv	Zustand des Slave Nur ein aktiver Slave kann mit einem PROFIBUS DP Master kommunizieren. Standardwert: Aktiviert
DPV0 Sync aktiv	Der *Sync Mode* ermöglicht ein zeitgleiches Aktivieren der Ausgänge mehrerer DP-V0-Slaves. **Achtung:** Bei DP-V2-Slaves, die im *Isochron Mode Sync* arbeiten, muss dieses Feld deaktiviert sein. Standardwert: Deaktiviert
DPV0 Freeze aktiv	Der *Freeze Mode* ermöglicht eine zeitgleiche Übernahme der Eingangsdaten mehrerer DP-V0-Slaves. **Achtung:** Bei DP-V2-Slaves, die im *Isochron Mode Freeze* arbeiten, muss dieses Feld deaktiviert sein. Standardwert: Deaktiviert
Watchdog aktiv	Wenn ausgewählt, erkennt der Slave den Ausfall eines Masters und geht in einen sicheren Zustand. Standardwert: Deaktiviert
Watchdog-Zeit [ms]	Kontrollkästchen Watchdog aktiv muss aktiviert sein. Wenn innerhalb dieser Zeitspanne kein Datenaustausch zwischen Master und Slave stattgefunden hat, schaltet sich der Slave ab und setzt alle DP-Output Daten auf ihren Initialwert zurück. 0 = Deaktiviert Richtwert: Watchdog-Zeit des Slaves > 6 * Ttr Wertebereich: 0 ... 65535 Standardwert: 0
Bei Ausfall letzte Daten senden	FALSE: Verbindung wird im Fehlerfall abgebaut und neu aufgebaut. TRUE: Sendet im Fehlerfall weiter Daten auch ohne Bestätigung des Slaves. Standardwert: Deaktiviert
Auto-Clear bei Ausfall	Ist *Auto-Clear bei Ausfall* im Master und bei diesem Slave auf TRUE gesetzt, dann schaltet der Master bei einem Ausfall dieses Slaves den kompletten PROFIBUS DP in den sicheren Zustand. Standardwert: Aktiviert

Tabelle 81: Register Parameter des PROFIBUS DP Slave

Register Gruppen

In diesem Register können die Slaves in verschiedenen Gruppen organisiert werden. Die GlobalControl-Kommandos *Sync* und *Freeze* können dann gezielt eine oder mehrere Gruppen ansprechen.

Element	Beschreibung	
Mitglied in Gruppe 1	Mitglied in Gruppe 1	Standardwert: Deaktiviert
Mitglied in Gruppe 2	Mitglied in Gruppe 2	
Mitglied in Gruppe 3	Mitglied in Gruppe 3	
Mitglied in Gruppe 4	Mitglied in Gruppe 4	
Mitglied in Gruppe 5	Mitglied in Gruppe 5	
Mitglied in Gruppe 6	Mitglied in Gruppe 6	
Mitglied in Gruppe 7	Mitglied in Gruppe 7	
Mitglied in Gruppe 8	Mitglied in Gruppe 8	

Tabelle 82: Register Gruppen im Eigenschaftendialog des PROFIBUS DP Slave

Register DP-V1

In diesem Register befinden sich Parameter, die erst ab DP-V1 definiert sind. Bei DP-V0-Slaves kann hier nichts ausgewählt werden. Welche Parameter von dem Slave unterstützt werden, erkennt man in der Spalte Unterstützt.

Element	Beschreibung
DP-V1	Wenn der DP-V1-Modus nicht aktiviert ist, können auch die anderen DP-V1-Features nicht genutzt werden. Der Slave verhält sich dann wie ein DP-V0-Slave. Eventuell müssen dann auch die Parametrierdaten geändert werden (siehe Handbuch des Slaves) Standardwert: Deaktiviert
Failsafe	Wenn dieser Modus aktiviert ist, sendet der Master im Zustand CLEAR keine Nullen als Ausgangsdaten zum Slave, sondern ein leeres Datenpaket (Failsafe-Paket). Der Slave erkennt daran, dass er jetzt die sicheren Ausgangsdaten (die nicht notwendigerweise alle Null sind) auf die Ausgänge legen soll. Standardwert: Deaktiviert
Isochron Modus	Diese Funktion ermöglicht eine taktsynchrone Regelung in Master und Slave unabhängig von der Belastung des Busses. Der Buszyklus wird mit einer Taktabweichung von < 1ms synchronisiert. Damit können hochgenaue Positionierungsvorgänge realisiert werden. Standardwert: Deaktiviert
Publisher aktiv	Diese Funktion wird für den Slave-Querverkehr benötigt. Dies ermöglicht die direkte und zeitsparende Kommunikation zwischen den Slaves via Broadcast ohne Umwege über den Master. Standardwert: Deaktiviert
Prm Block Struct. Supp.	Der Slave unterstützt strukturierte Parametrierdaten (Nur Lesen). Standardwert: Deaktiviert
Check. Cfg.-Mode	Reduzierte Konfigurationskontrolle, wenn Check Cfg.-Mode aktiviert ist, dann kann der Slave ohne die komplette Konfiguration betrieben werden. Für die Inbetriebnahme sollte dieses Feld deaktiviert werden. Standardwert: Deaktiviert

Tabelle 83: Register DP-V1 im Eigenschaftendialog des PROFIBUS DP Slave

Register Alarme

Auf dieser Seite können Alarme aktiviert werden. Das geht jedoch nur bei DP-V1-Slaves, wenn DP-V1 aktiviert ist und der Slave Alarme unterstützt. Welche Alarme unterstützt werden, erkennt man an dem Häkchen in der Spalte **Unterstützt**. Wird ein Alarm vorgeschrieben, erkennt man dies in der Spalte **Verlangt**.

Element	Beschreibung	
Update-Alarm	Alarm, wenn Parameter eines Moduls geändert.	Standardwert: Deaktiviert
Status-Alarm	Alarm, wenn Zustand eines Moduls geändert.	
Vendor-Alarm	Herstellerspezifischer Alarm.	
Diagnose-Alarm	Alarm bei bestimmten Ereignissen wie Kurzschluss, Übertemperatur etc. an einem Modul.	
Prozess-Alarm	Alarm bei wichtigen Ereignissen im Prozess.	
Stecken/Ziehen-Alarm	Alarm bei Ziehen oder Stecken eines Moduls.	

Tabelle 84: Register Alarme im Eigenschaftendialog des PROFIBUS DP Slave

Register Daten

In diesem Register befinden sich Informationen über die unterstützten Datenlängen, sowie über die Benutzerdaten (erweiterte Parametrierdaten).

Element	Beschreibung
Max. Input Len	Maximale Länge der Eingangsdaten.
Max. Output Len	Maximale Länge der Ausgangsdaten.
Max. Data Len	Maximale Gesamtlänge der Ein- und Ausgangsdaten.
Benutzerdatengrösse	Länge der Benutzerdaten.
Benutzerdaten	Parametrierdaten. Das Editieren empfiehlt sich nicht hier durchzuführen. Komfortabler geht es mit dem Dialog *User Parameter* bearbeiten Kapitel 6.8.
Max. Diag. Data Len	Maximale Länge der Diagnosedaten, die der Slave sendet.

Tabelle 85: Register Daten im Eigenschaftendialog des PROFIBUS DP Slave

Register Modell

Auf dieser Seite befinden sich verschiedene Informationen, die selbsterklärend sind:

Element	Beschreibung
Modell	Herstellerbezeichnung des PROFIBUS DP Slave
Hersteller	Hersteller des Feldgerätes
Ident. Nummer	Slave-Kennung der PROFIBUS Nutzerorganisation (PNO)
Revision	Ausgabestand des PROFIBUS DP Slave
Hardware-Release	Hardware-Ausgabestand des PROFIBUS DP Slave
Software-Release	Software-Ausgabestand des PROFIBUS DP Slave
GSD-Dateiname	Dateiname der GSD-Datei
Infotext	Zusätzliche Info zum PROFIBUS DP Slave

Tabelle 86: Register Modell im Eigenschaftendialog des PROFIBUS DP Slave

Register Features

Element	Beschreibung
Modularstation	TRUE: Modularstation FALSE: Kompaktstation
Erste parametrierbare Slotnummer	Nummerierung der Module (Slots) muss mit dieser Nummer beginnen und fortlaufend erfolgen.
Max. Module	Maximale Anzahl an Modulen, die eine modulare Station aufnehmen kann.
Unterstützung für 'Set Slave Add'	Slave unterstützt dynamische Adressvergabe.
Min. Slave Intervall [ms]	Die Mindestzeit, die zwischen zwei zyklischen Aufrufen des Slaves verstreichen muss.
Diag. Update	Anzahl der Pollzyklen, bis die Diagnose des Slaves den aktuellen Zustand widerspiegelt.
Unterstützung für WDBase1ms	Slave unterstützt 1ms als Zeitbasis für die Watchdog.
Unterstützung für DPV0 Sync	Slave unterstützt DP-V0 Sync
Unterstützung für DPV0 Freeze	Slave unterstützt DP-V0 Freeze
DPV1 Datentypen	Slave unterstützt die DP-V1-Datentypen.
Extra Alarm SAP	Slave unterstützt SAP 50 zur Alarmbestätigung.
Anzahl paralleler, aktiver Alarme	Gibt an, wie viele aktive Alarme der Slave gleichzeitig bearbeiten kann. Null bedeutet ein Alarm von jedem Typ.

Tabelle 87: Register Features im Eigenschaftendialog des PROFIBUS DP Slave

Register Übertragungsraten

In diesem Register befinden sich die Übertragungsraten, die der Slave unterstützt, sowie jeweils die zugehörige *MaxTsdr*.

MaxTsdr ist die Zeit, innerhalb welcher der Slave spätestens auf eine Anforderung vom Master antworten muss. Der Wertebereich ist abhängig vom Slave und der Übertragungsgeschwindigkeit und liegt zwischen 15 und 800 Tbit.

Element	Beschreibung
9,6k	MaxTsdr = 60
19,2k	MaxTsdr = 60
31,25k	Wird nicht unterstützt
45,45k	MaxTsdr = 60
93,75k	MaxTsdr = 60
187,5k	MaxTsdr = 60
500k	MaxTsdr = 70
1,5M	MaxTsdr = 75
3M	MaxTsdr = 90
6M	MaxTsdr = 100
12M	MaxTsdr = 120

Tabelle 88: Register Übertragungsraten im Eigenschaftendialog des PROFIBUS DP Slave

Register Azyklisch

In diesem Register befinden sich einige Parameter für die azyklische Datenübertragung:

Element	Beschreibung
C1 Read/Write-Unterstützung	Slave unterstützt azyklische Datenübertragung.
C1 Read/Write-notwendig	Slave erfordert azyklische Datenübertragung.
C1 Datengröße [Byte]	Maximale Länge eines azyklischen Datenpaketes.
C1 Response Timeout [ms]	Timeout für azyklische Datenübertragung.

Tabelle 89: Register Azyklisch im Eigenschaftendialog des PROFIBUS DP Slave

6.7 GSD-Datei einlesen

Die GSD-Datei enthält die Daten für die Parametrierung des PROFIBUS DP Slave.

So lesen Sie die GSD-Datei für den neuen PROFIBUS DP Slave ein:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master**, **PROFIBUS Slave** öffnen.
2. Im Kontextmenü vom PROFIBUS DP Master **GSD-Datei einlesen** wählen und die zum PROFIBUS Slave zugehörige GSD-Datei (z. B. hax100ea.gsd) auswählen.

i Die GSD-Dateien für HIMax Steuerungen finden Sie auf der HIMA Webseite www.hima.com.

Für die Richtigkeit der GSD-Datei ist der Hersteller des jeweiligen Feldgerätes verantwortlich.

Die GSD-Datei *hax100ea.gsd* des HIMax PROFIBUS DP Slave stellt die folgenden Module bereit:

PROFIBUS DP Master Eingangs Module	Typ	Anzahl
DP-Input/ELOP-Export	Byte	1
DP-Input/ELOP-Export	Bytes	2
DP-Input/ELOP-Export	Bytes	4
DP-Input/ELOP-Export	Bytes	8
DP-Input/ELOP-Export	Bytes	16
DP-Input/ELOP-Export	Word	1
DP-Input/ELOP-Export	Words	2
DP-Input/ELOP-Export	Words	4
DP-Input/ELOP-Export	Words	8
DP-Input/ELOP-Export	Words	16
PROFIBUS DP Master Ausgangs Module	Typ	Anzahl
DP-Output/ELOP-Import	Byte	1
DP-Output/ELOP-Import	Bytes	2
DP-Output/ELOP-Import	Bytes	4
DP-Output/ELOP-Import	Bytes	8
DP-Output/ELOP-Import	Bytes	16
DP-Output/ELOP-Import	Word	1
DP-Output/ELOP-Import	Words	2
DP-Output/ELOP-Import	Words	4
DP-Output/ELOP-Import	Words	8
DP-Output/ELOP-Import	Words	16

Tabelle 90: GSD-Datei des HIMax PROFIBUS DP Slave

6.8 User-Parameter bearbeiten

In dem Benutzerdatenfeld werden die **Startadresse** und die **Anzahl der Variablen** der Blöcke definiert.

Zusätzlich muss der PROFIBUS DP Master auch für die Anzahl der tatsächlich zu übertragenden Bytes konfiguriert werden. Dies geschieht durch Auswahl der PROFIBUS DP Module aus der GSD-Datei des PROFIBUS DP Slave (siehe auch Kapitel 6.2.2).

So öffnen Sie den Dialog User-Parameter bearbeiten:

1. Im Strukturbaum **Konfiguration**/**Ressource**/**Protokolle PROFIBUS DP Master** öffnen.
2. Rechtsklick auf **PROFIBUS Slave** und **Eigenschaften** wählen.
3. Auf Register **Daten** klicken und **Schaltfläche ...** neben den Benutzerdaten wählen.

i Der Aufbau des Dialogs **User Parameter bearbeiten** wird von der GSD-Datei des Slaves festgelegt.

Prinzipieller Aufbau des 32 Byte Benutzerdatenfelds:

Das 32 Byte Benutzerdatenfeld ist wie folgt aufgebaut:

Die 32 Byte sind in **acht Blöcke gruppiert**, mit jeweils **vier Bytes per Block**.

Die Blöcke 1...4 definieren welche und wie viele Variablen der PROFIBUS DP Master vom PROFIBUS DP Slave empfängt.

Die Blöcke 5...8 definieren welche und wie viele Variablen der PROFIBUS DP Master an den PROFIBUS DP Slave sendet.

Die ersten beiden Bytes eines jeden Blocks spezifizieren die Startadresse für die erste zu lesende oder zu schreibende Variable eines Blocks.

Die letzten beiden Bytes eines jeden Blocks spezifizieren die Anzahl der Variablen, die empfangen oder gesendet werden sollen.

Konfiguration der Benutzerdaten in verschiedenen Blöcken:

Normalerweise ist es nicht notwendig, die Variablen (Benutzerdaten) auf verschiedene Blöcke zu verteilen. Es ist vollkommen ausreichend nur den jeweils ersten Variablenblock der Eingangs und Ausgangsvariablen zu definieren und die Daten *en bloc* zu lesen und zu schreiben.

In Anwendungen, in denen es jedoch erforderlich ist, nur ausgewählte Variablen zu lesen und zu schreiben, können bis zu je vier Variablenblöcke für die Ausgangs- und Eingangsvariablen definiert werden.

Beispiel

Der PROFIBUS DP Master sendet und empfängt die folgenden Variablen vom PROFIBUS DP Slave:

1. Block: **4** Eingangsvariablen ab der Startadresse **0**.

2. Block: **6** Eingangsvariablen ab der Startadresse **50**.

4. Block :**9** Eingangsvariablen ab der Startadresse **100**.

5. Block: **2** Ausgangsvariablen ab der Startadresse **10**.

Konfiguration der Benutzerdaten im PROFIBUS DP Master:

Master Import/Slave Export	Startadresse	Anz. Variable
1. Block (Byte 0 ... 3)	0,0	0,4
2. Block (Byte 4 ... 7)	0,50	0,6
3. Block (Byte 8 ... 11)	0,0	0,0
4. Block (Byte 12 ... 15)	0,100	0,9

Tabelle 91: Beispiel: Blöcke 1 ... 4 des Benutzerdatenfeldes

Master Export/Slave Import	Startadresse	Anz. Variable
5. Block (Byte 16 ... 19)	0,10	0,2
6. Block (Byte 20 ... 23)	0,0	0,0
7. Block (Byte 24 ... 27)	0,0	0,0
8. Block (Byte 28 ... 31)	0,0	0,0

Tabelle 92: Beispiel: Blöcke 1 ... 4 des Benutzerdatenfeldes

Dialog *User Parameter bearbeiten* eines HIMatrix oder HIMax PROFIBUS DP Slaves.

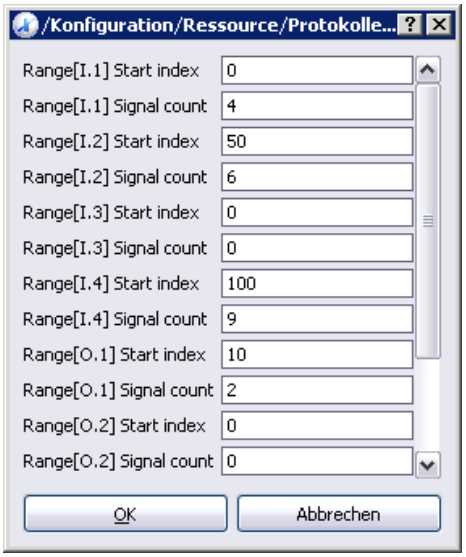


Bild 34: Dialog *User Parameter bearbeiten*

6.9 PROFIBUS Funktionsbausteine

Mit den PROFIBUS Funktionsbausteinen können Sie den HIMA PROFIBUS DP Master und die ihm zugeordneten PROFIBUS DP Slaves optimal den Erfordernissen Ihres Projekts anpassen.

Die Funktionsbausteine parametrieren Sie im Anwenderprogramm, sodass die Funktionen des Masters und der Slaves (Alarme, Diagnosedaten, Zustände) im Anwenderprogramm gesetzt und gelesen werden können.

i Funktionsbausteine werden nur für spezielle Anwendungen benötigt. Für den normalen zyklischen Datenverkehr zwischen Master und Slave sind diese Funktionsbausteine nicht erforderlich!

Prinzipielle Konfiguration der PROFIBUS DP Funktionsbausteine, siehe Kapitel 12.1.

Folgende Funktionsbausteine stehen zur Verfügung:

Funktionsbaustein	Beschreibung der Funktion	Geeignet ab DP Leistungsstufe
MSTAT 6.9.1	Zustand des Master durch das Anwenderprogramm steuern	DP-V0
RALRM 6.9.2	Alarmmeldungen der Slaves lesen	DP-V1
RDIAG 6.9.3	Diagnosemeldungen der Slaves lesen	DP-V0
RDREC 6.9.4	Azyklische Datensätze der Slaves lesen	DP-V1
SLACT 6.9.5	Zustand der Slaves durch das Anwenderprogramm steuern	DP-V0
WRREC 6.9.6	Azyklische Datensätze der Slaves schreiben	DP-V1

Tabelle 93: Übersicht PROFIBUS DP Funktionsbausteine

i HIMA PROFIBUS DP Master arbeiten mit der Leistungsstufe DP-V1.

HIMA PROFIBUS DP Slaves arbeiten mit der Leistungsstufe DP-V0.

Beachten Sie, dass dadurch nicht alle Funktionsbausteine des HIMA PROFIBUS DP Masters einen HIMA PROFIBUS DP Slave steuern können.

6.9.1 Funktionsbaustein MSTAT

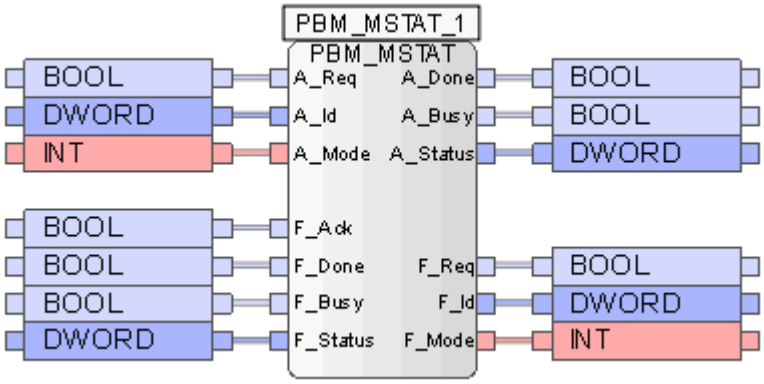


Bild 35: Funktionsbaustein MSTAT

Mit dem Funktionsbaustein **MSTAT** (ab DP-V0) kann der PROFIBUS DP Master vom Anwenderprogramm gesteuert werden. Somit ist es möglich, den PROFIBUS DP Master durch einen mechanischen Schalter an einem physikalischen Eingang oder durch einen Timer in einen der folgenden Zustände zu setzen:

- 0: OFFLINE
- 1: STOP
- 2: CLEAR
- 3: OPERATE

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Baustein	BOOL
A_ID	Master ID (nicht genutzt)	DWORD
A_Mode	In folgende Zustände kann der PROFIBUS DP Master gesetzt werden 0: OFFLINE 1: STOP 2: CLEAR 3: OPERATE	INT

Tabelle 94: A-Eingänge Funktionsbaustein MSTAT

A-Ausgänge	Beschreibung	Typ
A_Done	TRUE: Der PROFIBUS DP Master wurde in den am Eingang A_Mode definierten Zustand gesetzt.	BOOL
A_Busy	TRUE: Das Setzen des PROFIBUS DP Master ist noch nicht beendet.	BOOL
A_Status	Status oder Fehlercode (siehe Kapitel 6.11)	DWORD

Tabelle 95: A-Ausgänge Funktionsbaustein MSTAT

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein MSTAT im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins MSTAT im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein MSTAT (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **MSTAT** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **MSTAT** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_ACK	BOOL
F_DONE	BOOL
F_BUSY	BOOL
F_STATUS	DWORD

Tabelle 96: F-Eingänge Funktionsbaustein MSTAT

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **MSTAT** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **MSTAT** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_REQ	BOOL
F_ID	DWORD
F_MODE	INT

Tabelle 97: F-Ausgänge Funktionsbaustein MSTAT

So erstellen Sie den Funktionsbaustein MSTAT im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **MSTAT** wählen.
3. Rechtsklick auf Funktionsbaustein **MSTAT** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **MSTAT** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **MSTAT** im Anwenderprogramm verbunden haben.

Eingänge	Typ
M_ID	DWORD
MODE	INT
REQ	BOOL

Tabelle 98: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **MSTAT** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **MSTAT** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
DONE	BOOL
STATUS	DWORD

Tabelle 99: Ausgangs-Systemvariablen

So bedienen Sie den Funktionsbaustein MSTAT:

1. Im Anwenderprogramm den Eingang *A_Mode* auf den gewünschten Zustand setzen. Wird *A_Mode* nicht gesetzt, wird nach Schritt 2 ein Fehlercode am Ausgang *A_Status* ausgegeben und der PROFIBUS DP Master wird nicht gesetzt.
2. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

☑ Der Ausgang *A_Busy* ist TRUE, bis der MSTAT-Befehl abgearbeitet ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Done* auf TRUE.

i Konnte der vorgegebene Mode nicht gesetzt werden, wird ein Fehlercode am Ausgang *A_Status* ausgegeben.

Der aktuelle Mode des Masters kann der Variablen Master-Status entnommen werden (siehe Kapitel 6.10).

6.9.2 Funktionsbaustein RALRM

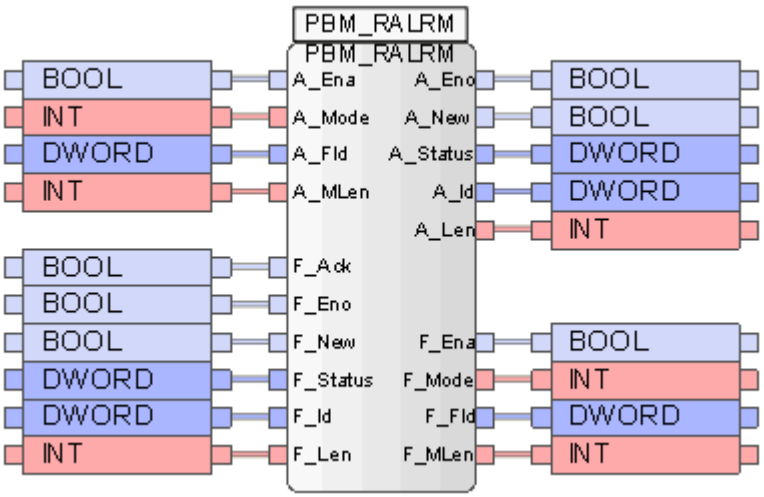


Bild 36: Funktionsbaustein RALRM

Der Funktionsbaustein **RALRM** (ab DP-V1) dient zur Auswertung von Alarmen.

Alarme sind eine spezielle Form von Diagnosemeldungen, die vorrangig behandelt werden. Alarme melden der Anwendung, wichtige Ereignisse, die Reaktionen seitens der Anwendung erfordern (z. B. ein WRREC). Dies ist jedoch herstellerabhängig und kann dem Gerätehandbuch des PROFIBUS DP Slaves entnommen werden.

Solange der Funktionsbaustein **RALRM** aktiv ist, wartet dieser auf Alarmmeldungen der Slaves. Wird ein Alarm empfangen, wird der Ausgang *A_NEW* für mindestens einen Zyklus auf TRUE geschaltet und die Alarmdaten können per Alarmtelegramm ausgelesen werden. Vor dem nächsten Alarm geht *A_NEW* für mindesten einen Zyklus auf FALSE. Alle Alarme werden implizit bestätigt. Es geht kein Alarm verloren.

Das Anwenderprogramm ist bei Verwendung mehrerer Funktionsbausteine **RALRM** so anzulegen, dass immer nur ein Funktionsbaustein **RALRM** aktiv ist.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Ena	Mit TRUE wird der Funktionsbaustein freigegeben	BOOL
A_Mode	Nicht genutzt	INT
A_FID	Nicht genutzt	DWORD
A_MLen	Maximal erwartete Länge der zu empfangenden Alarmdaten in Bytes	INT

Tabelle 100: A-Eingänge Funktionsbaustein RALRM

A-Ausgänge	Beschreibung	Typ
A_Eno	TRUE: Funktionsbaustein aktiv FALSE: Funktionsbaustein nicht aktiv	BOOL
A_New	TRUE: Neuer Alarm wurde empfangen FALSE: Kein neuer Alarm	BOOL
A_Status	Status oder Fehlercode (siehe Kapitel 6.11)	DWORD
A_ID	Identifikationsnummer des Alarm auslösenden Slave	DWORD
A_Len	Länge der empfangenen Alarmdaten in Bytes	INT

Tabelle 101: A-Ausgänge Funktionsbaustein RALRM

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein RALRM im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **RALRM** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **RALRM** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **RALRM** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **RALRM** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_ACK	BOOL
F_ENO	BOOL
F_NEW	BOOL
F_STATUS	DWORD
F_ID	DWORD
F_LEN	INT

Tabelle 102: F-Eingänge Funktionsbaustein RALRM

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **RALRM** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **RALRM** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Ena	BOOL
F_MODE	INT
F_FID	DWORD
F_MLEN	INT

Tabelle 103: F-Ausgänge Funktionsbaustein RALRM

So erstellen Sie den Funktionsbaustein RALRM im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **RALRM** wählen.
3. Rechtsklick auf Funktionsbaustein **RALRM** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **RALRM** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **RALRM** im Anwenderprogramm verbunden haben.

Eingänge	Typ
EN	BOOL
F_ID	DWORD
MLEN	INT
MODE	INT

Tabelle 104: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **RALRM** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **RALRM** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
ENO	BOOL
ID	DWORD
LEN	INT
NEW	BOOL
STATUS	DWORD

Tabelle 105: Ausgangs-Systemvariablen

Im Register „Prozessvariablen“ des Funktionsbausteins **RALRM** im Strukturbaum sind Variablen zu definieren, deren Struktur zu den Alarmdaten passt. Werden keine Variablen definiert, können Alarmdaten zwar angefordert, aber nicht gelesen werden.

Eine Alarmmeldung enthält mindestens vier Bytes. Die ersten vier Bytes der Alarmmeldung enthalten die Standard-Alarmdaten.

Zur vereinfachten Auswertung der Standard-Alarme stellt HIMA den Hilfsfunktionsbaustein **ALARM** (siehe Kapitel 6.10) bereit. Wenn Sie diesen Baustein verwenden wollen, fassen Sie die ersten vier Bytes in einer Variablen vom Typ DWORD zusammen und geben Sie diese Variable auf den Eingang *IN* des Hilfsfunktionsbausteins **ALARM**.

i Enthält ein Alarmtelegramm mehr Bytes als im Register „Daten“ definiert wurden, wird nur die Anzahl der definierten Bytes übernommen. Der Rest wird abgeschnitten.

Alarmdaten	Beschreibung
Byte 0	Länge der Alarmmeldung in Byte (4 ... 126)
Byte 1	Kennung für den Alarmtyp 1: Diagnosealarm 2: Prozessalarm 3: Ziehenalarm 4: Steckenalarm 5: Statusalarm 6: Updatealarm 31: Ausfall einer Erweiterung eines Master oder Slaves 32 ... 126: Herstellerspezifisch: Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden.
Byte 2	Steckplatznummer der Alarm auslösenden Komponente
Byte 3	0:keine weitere Information 1:ankommender Alarm, Slot gestört 2:ausgehender Alarm, Slot nicht mehr gestört 3:ausgehender Alarm, Slot weiterhin gestört
Byte 4 bis 126	Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden

Tabelle 106: Alarmdaten

i Die Struktur der Standard-Alarme (Bytes 0…3) ist normiert und für alle Hersteller identisch. Für die herstellerspezifisch genutzten Bytes 4…126 schlagen Sie im Gerätehandbuch des PROFIBUS DP Slave nach.

Beachten Sie, dass Geräte nach dem DP-V0-Standard keine Alarmtelegramme unterstützen.

So bedienen Sie den Funktionsbaustein RALRM:

1. Im Anwenderprogramm am Eingang *A_Mlen* die Anzahl der maximal zu erwartetenden Alarmdaten in Bytes definieren. Während des Betriebs kann *A_Mlen* nicht geändert werden.
2. Im Anwenderprogramm den Eingang *A_Ena* auf TRUE setzen.

i Im Gegensatz zu den anderen Funktionsbausteinen, ist der Funktionsbaustein **RALRM** nur aktiv, solange der Eingang *A_Ena* TRUE ist.

Wurde der Baustein erfolgreich gestartet, dann geht der Ausgang *A_Eno* auf TRUE. Konnte der Baustein nicht gestartet werden, wird ein Fehlercode am Ausgang *A_Status* ausgegeben.

Trifft ein neuer Alarm ein, geht der Ausgang *A_New* für mindestens einen Zyklus auf TRUE. Für diese Zeit enthalten die Ausgänge die Alarmdaten des Alarm auslösenden Slaves, die ausgewertet werden können.

Danach geht der Ausgang *A_New* wieder für mindestens einen Zyklus auf FALSE. Die Ausgänge *A_ID* und *A_Len* werden auf Null zurückgesetzt, bevor die nächste Alarmmeldung empfangen und ausgewertet werden kann.

6.9.3 Funktionsbaustein RDIAG

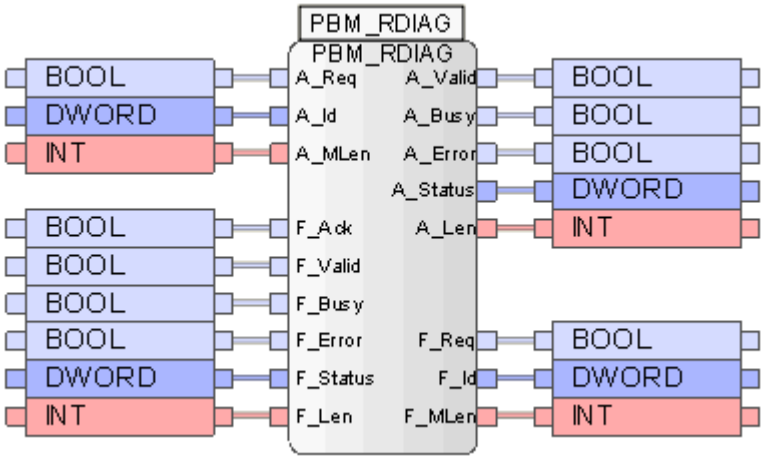


Bild 37: Funktionsbaustein RDIAG

Der Funktionsbaustein **RDIAG** (ab DP-V0) dient zum Auslesen der aktuellen Diagnosemeldung (6 Byte ... 240 Bytes) eines Slave.

Im HIMA PROFIBUS DP Master dürfen beliebig viele **RDIAG**-Bausteine gleichzeitig aktiv sein.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet die Anforderung einer Diagnosemeldung	BOOL
A_ID	Identifikationsnummer des Slave (siehe Kapitel 6.10)	DWORD
A_MLen	Maximal erwartete Länge der zu lesenden Diagnosemeldung in Bytes	INT

Tabelle 107: A-Eingänge Funktionsbaustein RALRM

A-Ausgänge	Beschreibung	Typ
A_Valid	Eine neue Diagnosemeldung wurde empfangen und ist gültig	BOOL
A_Busy	TRUE: Das Lesen ist noch nicht beendet	BOOL
A_Error	TRUE: Beim Lesen trat ein Fehler auf	BOOL
A_Status	Status oder Fehlercode (siehe Kapitel 6.11)	DWORD
A_Len	Länge der gelesenen Diagnosedaten in Bytes	INT

Tabelle 108: A-Ausgänge Funktionsbaustein RALRM

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **RDIAG** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **RDIAG** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **RDIAG** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **RDIAG** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **RDIAG** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_ACK	BOOL
F_VALID	BOOL
F_BUSY	BOOL
F_ERROR	BOOL
F_Status	DWORD
F_LEN	INT

Tabelle 109: F-Eingänge Funktionsbaustein RDIAG

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **RDIAG** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **RDIAG** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD
F_MLen	INT

Tabelle 110: F-Ausgänge Funktionsbaustein RDIAG

So erstellen Sie den Funktionsbaustein RDIAG im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **RDIAG** wählen.
3. Rechtsklick auf Funktionsbaustein **RDIAG** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **RDIAG** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **RDIAG** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
MLEN	INT
REQ	BOOL

Tabelle 111: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **RDIAG** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **RDIAG** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
ERROR	BOOL
LEN	INT
Status	DWORD
VALID	BOOL

Tabelle 112: Ausgangs-Systemvariablen

Diagnosedaten

Im Register **Daten** sind Variablen zu definieren, deren Struktur zu den Diagnosedaten passen muss. Eine Diagnosemeldung enthält mindestens sechs Bytes und maximal 240 Bytes. Die ersten vier Bytes der Diagnosemeldung enthalten die Standard-Diagnose.

Zur vereinfachten Auswertung der Standarddiagnose stellt HIMA den Hilfsfunktionsbaustein **STDDIAG** (siehe Kapitel 6.10) bereit. Wenn Sie diesen Baustein verwenden wollen, fassen Sie die ersten vier Bytes in einer Variablen vom Typ DWORD zusammen und geben Sie diese Variable auf den Eingang *IN* des Hilfsfunktionsbausteins **STDDIAG**.

i Enthält ein Diagnosetelegramm mehr Bytes als im Register „Daten“ definiert wurden, wird nur die Anzahl der definierten Bytes übernommen. Der Rest wird abgeschnitten

Diagnosedaten	Beschreibung
Byte 0	Byte 0 ... 3 enthalten die Standarddiagnose. Die Standarddiagnose kann als Variable vom Typ DWORD mit dem Hilfsfunktionsbaustein **STDDIAG** dekodiert werden.
Byte 1	
Byte 2	
Byte 3	Busadresse des Masters, dem ein Slave zugeordnet ist.
Byte 4	High-Byte (Herstellerkennung)
Byte 5	Low-Byte (Herstellerkennung)
Byte 6 ... 240	Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden

Tabelle 113: Diagnosedaten

i Die HIMA Slaves liefern ein Diagnosetelegramm von sechs Bytes Länge. Die Bedeutung der Bytes ist standardisiert.

Für Slaves anderer Hersteller sind nur die ersten sechs Bytes funktionell identisch.

Für weitere Informationen über das Diagnosetelegramm beachten Sie die Herstellerbeschreibung des Slaves.

So bedienen Sie den Funktionsbaustein RDIAG:

1. Im Anwenderprogramm die Slaveadresse an den Eingang *A_ID* setzen.
2. Im Anwenderprogramm am Eingang *A_Mlen* die Anzahl der maximal zu erwartetenden Diagnosedaten in Bytes definieren.
3. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis die Diagnoseanforderung abgearbeitet ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Valid* oder *A_Error* auf TRUE.

Ist das Diagnosetelegramm gültig, geht der Ausgang *A_Valid* auf TRUE. Die Diagnosedaten können über die im Register Daten definierten Variablen ausgewertet werden. Der Ausgang *A_Len* enthält die Anzahl der Diagnosedaten in Bytes, die tatsächlich ausgelesen wurden.

Konnte das Diagnosetelegramm nicht erfolgreich gelesen werden, dann wechselt der Ausgang *A_Error* auf TRUE und ein Fehlercode wird am Ausgang *A_Status* ausgegeben.

6.9.4 Funktionsbaustein RDREC

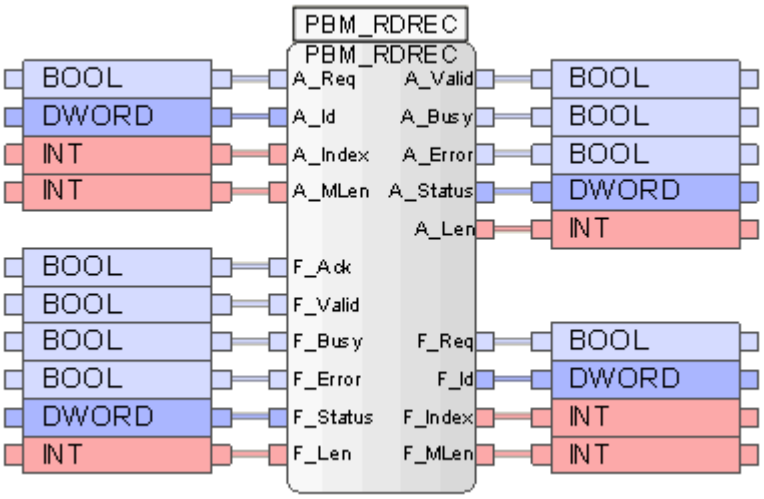


Bild 38: Funktionsbaustein RDREC

Der Funktionsbaustein **RDREC** dient zum azyklischen Lesen eines am Eingang *A_Index* adressierten Datensatzes von einem Slave. Welche Daten gelesen werden können, muss der Betriebsanleitung des Slaves entnommen werden.

Diese Funktion ist erst ab DP-V1 definiert und optional.

Im HIMA PROFIBUS DP Master können gleichzeitig bis zu 32 **RDREC**- und/oder **WRREC**-Bausteine aktiv sein.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet die Anforderung zum Lesen.	BOOL
A_Id	Identifikationsnummer des Slave, siehe Kapitel 6.10	DWORD
A_Index	Datensatznummer des zu lesenden Datensatzes. Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden.	INT
A_MLen	Maximale Länge der zu lesenden Daten in Bytes.	INT

Tabelle 114: A-Eingänge Funktionsbaustein RDREC

A-Ausgänge	Beschreibung	Typ
A_Valid	Ein neuer Datensatz wurde empfangen und ist gültig.	BOOL
A_Busy	TRUE: Der Lesevorgang ist noch nicht beendet.	BOOL
A_Error	TRUE: Ein Fehler ist aufgetreten FALSE: Kein Fehler	BOOL
A_Status	Status oder Fehlercode, siehe Kapitel 6.11	DWORD
A_Len	Länge der gelesenen Datensatzinformation in Bytes.	INT

Tabelle 115: A-Ausgänge Funktionsbaustein RDREC

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **RDREC** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **RDREC** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein RDREC (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **RDREC** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **RDREC** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Valid	BOOL
F_Busy	BOOL
F_Error	BOOL
F_Status	DWORD
F_Len	INT

Tabelle 116: F-Eingänge Funktionsbaustein RDREC

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **RDREC** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **RDREC** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD
F_Index	INT
F_Mlen	INT

Tabelle 117: F-Ausgänge Funktionsbaustein RDREC

So erstellen Sie den Funktionsbaustein RDREC im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **RDREC** wählen.
3. Rechtsklick auf Funktionsbaustein **RDREC** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **RDREC** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **RDREC** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
INDEX	INT
MLEN	INT
REQ	BOOL

Tabelle 118: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **RDREC** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **RDREC** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
ERROR	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Tabelle 119: Ausgangs-Systemvariablen

Daten	Beschreibung
Es sind keine Variablen vorgegeben	Im Register *Prozessvariablen* kann eine beliebige Datenstruktur definiert werden, die allerdings auf die Struktur des Datensatzes passen muss. Die Struktur des Datensatzes muss aus der Bedienungsanleitung vom Hersteller des Slaves entnommen werden.

Tabelle 120: Daten

So bedienen Sie den Funktionsbaustein RDREC:

1. Im Anwenderprogramm die Slaveadresse an den Eingang *A_ID* setzen.
2. Im Anwenderprogramm den Slave-spezifischen Index für den Datensatz (Handbuch des Herstellers) am Eingang *A_Index* setzen.
3. Im Anwenderprogramm die Länge des zu lesenden Datensatzes am Eingang *A_Len* setzen.
4. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis die Datensatzanforderung abgearbeitet ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Valid* oder *A_Error* auf TRUE.

Ist der Datensatz gültig, geht der Ausgang *A_Valid* auf TRUE. Der Datensatz kann über die im Register Daten definierten Variablen ausgewertet werden. Der Ausgang *A_Len* enthält die tatsächliche Länge des ausgelesenen Datensatzes.

Konnte der Datensatz nicht erfolgreich gelesen werden, dann wechselt der Ausgang *A_Error* auf TRUE und ein Fehlercode wird am Ausgang *A_Status* ausgegeben.

6.9.5 Funktionsbaustein SLACT

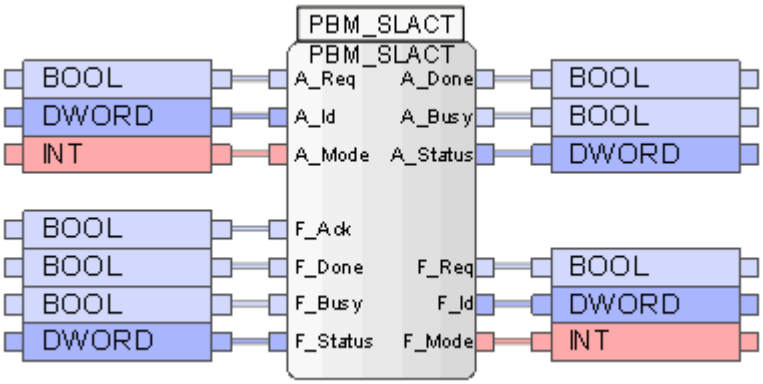


Bild 39: Funktionsbaustein SLACT

Der Funktionsbaustein **SLACT** (ab DP-V0) dient zum Aktivieren und Deaktivieren eines Slave aus dem Anwenderprogramm des PROFIBUS DP Master. Somit ist es möglich, den Slave durch einen mechanischen Schalter an einem physikalischen Eingang des PROFIBUS DP Master, oder durch einen Timer in einen der folgenden Zustände zu setzen.

≠ 0: Aktiv

= 0: Nicht aktiv

Das Anwenderprogramm ist bei Verwendung mehrerer Funktionsbausteine **SLACT** so anzulegen, dass immer nur ein Funktionsbaustein **SLACT** aktiv ist.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Baustein	BOOL
A_Id	Identifikationsnummer des Slave (siehe Kapitel 6.10)	DWORD
A_Mode	Zustand, in den der PROFIBUS DP Slave gesetzt werden soll: ≠ 0: Aktiv (Verbunden) = 0: Nicht aktiv (Deaktiviert)	INT

Tabelle 121: A-Eingänge Funktionsbaustein SLACT

A-Ausgänge	Beschreibung	Typ
A_Done	TRUE: Der PROFIBUS DP Slave wurde in den am Eingang „A_Mode“ definierten Zustand gesetzt.	BOOL
A_Busy	TRUE: Das Setzen des PROFIBUS DP Slave ist noch nicht beendet.	BOOL
A_Status	Status oder Fehlercode (siehe Kapitel 6.11)	DWORD

Tabelle 122: A-Ausgänge Funktionsbaustein SLACT

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **SLACT** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins SLACT im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein SLACT (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **SLACT** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **SLACT** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Done	BOOL
F_Busy	BOOL
F_Status	DWORD

Tabelle 123: F-Eingänge Funktionsbaustein SLACT

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **SLACT** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **SLACT** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD
F_Mode	INT

Tabelle 124: F-Ausgänge Funktionsbaustein SLACT

So erstellen Sie den Funktionsbaustein SLACT im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **SLACT** wählen.
3. Rechtsklick auf Funktionsbaustein **SLACT** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **SLACT** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **SLACT** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
MODE	INT
REQ	BOOL

Tabelle 125: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **SLACT** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **SLACT** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
DONE	BOOL
STATUS	DWORD

Tabelle 126: Ausgangs-Systemvariablen

So bedienen Sie den Funktionsbaustein SLACT:

1. Im Anwenderprogramm den Eingang *A_Mode* auf den gewünschten Zustand setzen.
2. Im Anwenderprogramm den Identifier mit der Slaveadresse am Eingang *A_ID* setzen.
3. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis der *SLACT*-Befehl abgearbeitet ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Done* auf TRUE.

Am Ausgang *A_Status* wird der Slave-Mode ausgegeben, wenn der Slave-Mode gesetzt werden konnte.

Am Ausgang *A_Status* wird ein Fehlercode ausgegeben, wenn der Slave-Mode nicht gesetzt werden konnte.

6.9.6 Funktionsbaustein WRREC

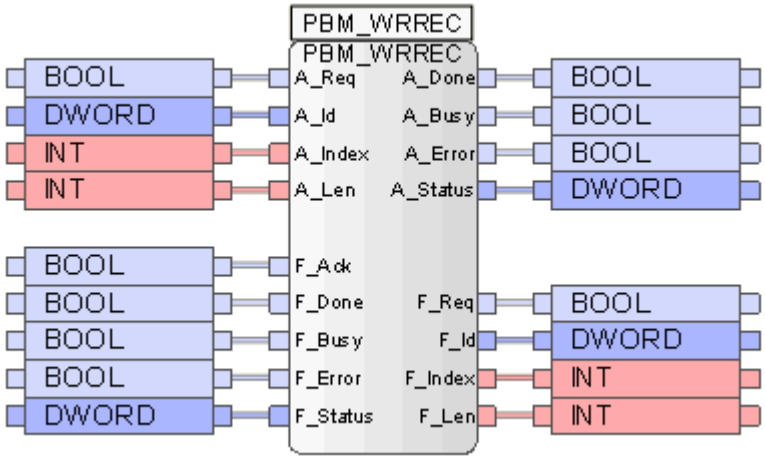


Bild 40: Funktionsbaustein WRREC

Der Funktionsbaustein **WRREC** (ab DP-V1) dient zum azyklischen Schreiben eines mit *A_Index* adressierten Datensatzes an einen Slave. Welche Daten geschrieben werden können, muss der Betriebsanleitung des Slaves entnommen werden.

Im HIMA PROFIBUS DP Master können gleichzeitig bis zu 32 **RDREC**- und/oder **WRREC**-Bausteine aktiv sein.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet die Anforderung zum Schreiben eines Datensatzes.	BOOL
A_ID	Identifikationsnummer des Slaves (siehe Kapitel 6.10)	DWORD
A_Index	Datensatznummer des zu schreibenden Datensatzes. Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden.	INT
A_Len	Länge des zu schreibenden Datensatzes in Bytes	INT

Tabelle 127: A-Eingänge Funktionsbaustein WRREC

A-Ausgänge	Beschreibung	Typ
A_Done	TRUE: Funktionsbaustein hat den Funktionsbaustein hat den Schreibvorgang beendet.	BOOL
A_Busy	TRUE: Funktionsbaustein hat den Schreibvorgang noch nicht beendet.	BOOL
A_Error	TRUE: Beim Schreibvorgang trat ein Fehler auf.	BOOL
A_STATUS	Status oder Fehlercode (siehe Kapitel 6.11)	DWORD

Tabelle 128: A-Ausgänge Funktionsbaustein WRREC

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **WRREC** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins WRREC im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein WRREC (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen Sie zuvor im Variableneditor erstellen.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **WRREC** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **WRREC** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Done	BOOL
F_Busy	BOOL
F_Error	BOOL
F_Status	DWORD

Tabelle 129: F-Eingänge Funktionsbaustein WRREC

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **WRREC** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **WRREC** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD
F_Index	INT
F_Len	INT

Tabelle 130: F-Ausgänge Funktionsbaustein WRREC

So erstellen Sie den Funktionsbaustein WRREC im Strukturbaum:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **PROFIBUS DP Master**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **WRREC** wählen.
3. Rechtsklick auf Funktionsbaustein **WRREC** und **Edit** wählen.
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **WRREC** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **WRREC** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
INDEX	INT
LEN	INT
REQ	BOOL

Tabelle 131: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **WRREC** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **WRREC** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
DONE	BOOL
ERROR	BOOL
STATUS	DWORD

Tabelle 132: Ausgangs-Systemvariablen

Daten	Beschreibung
Es sind keine Variablen vorgegeben	Im Register *Prozessvariablen* kann eine beliebige Datenstruktur definiert werden, die allerdings auf die Struktur des Datensatzes passen muss. Die Struktur des Datensatzes muss aus der Bedienungsanleitung vom Hersteller des Slaves entnommen werden.

Tabelle 133: Daten

Für die Bedienung des Funktionsbausteins WRREC sind die folgenden Schritte erforderlich:

1. Im Anwenderprogramm die Slaveadresse an den Eingang *A_ID* setzen.
2. Im Anwenderprogramm den Slave-spezifischen Index für den Datensatz (Handbuch des Herstellers) am Eingang *A_Index* setzen.
3. Im Anwenderprogramm die Länge des zu schreibenden Datensatzes am Eingang *A_Len* setzen.
4. Im Anwenderprogramm den Datensatz, wie im Register „Daten" definiert, einstellen.
5. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis der Datensatz geschrieben ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Done* auf TRUE.

Konnte der Datensatz nicht erfolgreich geschrieben werden, dann wechselt der Ausgang *A_Error* auf TRUE und ein Fehlercode wird am Ausgang *A_Status* ausgegeben.

6.10 PROFIBUS Hilfsfunktionsbausteine

Die Hilfsfunktionsbausteine verwenden Sie zur Parametrierung und Auswertung der Ein- und Ausgänge der Funktionsbausteine.

Folgende Hilfsfunktionsbausteine stehen zur Verfügung:

Hilfsfunktionsbausteine	Beschreibung der Funktion
ACTIVE (siehe Kap 6.10.1)	Ist der Slave Aktiv oder Inaktiv
ALARM (siehe Kap. 6.10.2)	Dekodieren der Alarmdaten
DEID (siehe Kap. 6.10.3)	Identifikationsnummer dekodieren
ID (siehe Kap. 6.10.4)	Die Funktion ID generiert aus vier Bytes einen Identifier
NSLOT (siehe Kap. 6.10.5)	Fortlaufende Identifikationsnummer für die Slots erstellen
SLOT (siehe Kap. 6.10.6)	SLOT Identifikationsnummer mit Slot-Nummer erstellen
STDDIAG (siehe Kap. 6.10.7)	Standarddiagnose eines Slaves dekodieren
LATCH	Wird nur innerhalb anderer Funktionsbausteine verwendet
PIG	Wird nur innerhalb anderer Funktionsbausteine verwendet
PIGII	Wird nur innerhalb anderer Funktionsbausteine verwendet

Tabelle 134: Übersicht Hilfsfunktionsbausteine

6.10.1 Hilfsfunktionsbaustein ACTIVE

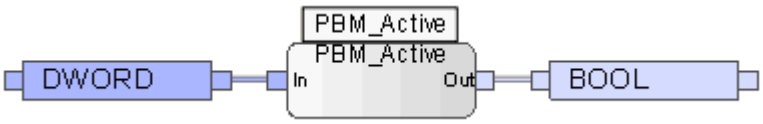


Bild 41: Hilfsfunktionsbaustein ACTIVE

Der Hilfsfunktionsbaustein ACTIVE ermittelt aus der Standarddiagnose eines PROFIBUS DP Slaves, ob der Slave gerade aktiv oder inaktiv ist.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
IN	Standarddiagnose des Slave	DWORD

Tabelle 135: Eingänge Hilfsfunktionsbaustein ACTIVE

Ausgänge	Beschreibung	Typ
OUT	TRUE: Slave ist aktiv FALSE: Slave ist inaktiv	BOOL

Tabelle 136: Ausgänge Hilfsfunktionsbaustein ACTIVE

6.10.2 Hilfsfunktionsbaustein ALARM

(Dekodieren der Alarmdaten)

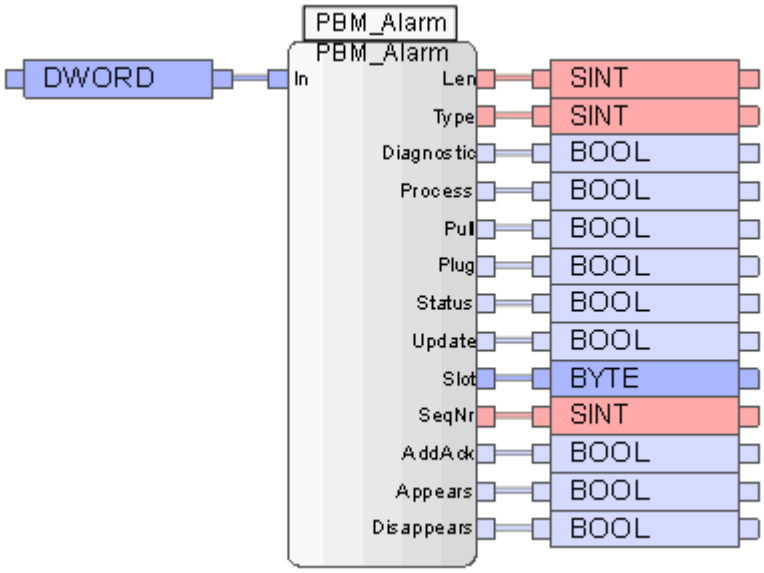


Bild 42: Hilfsfunktionsbaustein ALARM

Der Hilfsfunktionsbaustein **ALARM** dekodiert die Standard-Alarmdaten eines PROFIBUS DP Slaves.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
IN	Standardalarm	DWORD

Tabelle 137: Eingänge Hilfsfunktionsbaustein ALARM

Ausgang	Beschreibung	Typ
Len	Länge der gesamten Alarmmeldung	SINT
Type	1: Diagnosealarm 2: Prozessalarm 3: Ziehenalarm 4: Steckenalarm 5: Statusalarm 6: Updatealarm Andere Nummern sind entweder reserviert oder herstellerspezifisch. Die Bedeutung muss der Herstellerbeschreibung des Geräts entnommen werden.	SINT
Diagnostic	True = Diagnosealarm	BOOL
Process	True = Prozessalarm	BOOL
Pull	True = Modul wurde gezogen	BOOL
Plug	True = Modul wurde wieder gesteckt	BOOL
Status	True = Status-Alarm	BOOL
Update	True = Update-Alarm	BOOL
Slot	Alarmauslösendes Modul	BYTE
SeqNr	Alarm-Sequenznummer	SINT

<table>
<tr><th>Ausgang</th><th>Beschreibung</th><th>Typ</th></tr>
<tr><td>AddAck</td><td>TRUE bedeutet, dass der Slave, der diesen Alarm ausgelöst hat, eine zusätzliche Bestätigung durch die Anwendung erwartet. Welche genau, muss dem Slave-Handbuch des Herstellers entnommen werden.</td><td>BOOL</td></tr>
<tr><td>Appears
Disappears</td><td>
<table>
<tr><th>Ausgang</th><th>Wert</th><th>Beschreibung</th></tr>
<tr><td>Appears</td><td>TRUE</td><td rowspan="2">Sind beide FALSE, dann ist bis zu diesem Zeitpunkt kein Fehler aufgetreten</td></tr>
<tr><td>Disappears</td><td>FALSE</td></tr>
<tr><td>Appears</td><td>TRUE</td><td rowspan="2">Ein Fehler ist aufgetreten und steht noch an.</td></tr>
<tr><td>Disappears</td><td>FALSE</td></tr>
<tr><td>Appears</td><td>TRUE</td><td rowspan="2">Ein Fehler war aufgetreten und verschwindet gerade.</td></tr>
<tr><td>Disappears</td><td>FALSE</td></tr>
<tr><td>Appears</td><td>TRUE</td><td rowspan="2">Sind beide TRUE, dann verschwindet der Fehler, der Slave ist aber weiterhin gestört.</td></tr>
<tr><td>Disappears</td><td>FALSE</td></tr>
</table>
</td><td>BOOL</td></tr>
</table>

Tabelle 138: Ausgänge Hilfsfunktionsbaustein ALARM

6.10.3 Hilfsfunktionsbaustein DEID

(Identifikationsnummer dekodieren)

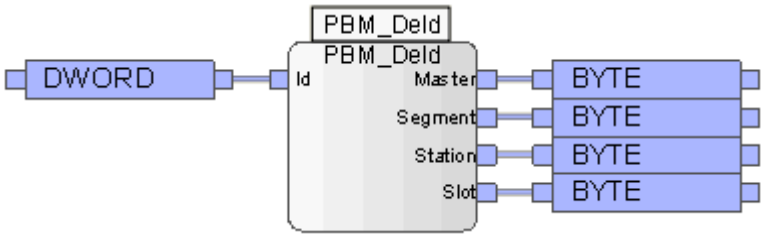


Bild 43: Hilfsfunktionsbaustein DEID

Der Hilfsfunktionsbaustein **DEID** dekodiert die Identifikationsnummer und zerlegt diese in ihre vier Bestandteile.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
Id	Identifikationsnummer des Slaves	DWORD

Tabelle 139: Eingänge Hilfsfunktionsbaustein DEID

Ausgänge	Beschreibung	Typ
Master	Busadresse des Master	BYTE
Segment	Segment	BYTE
Station	Busadresse des Slave	BYTE
Slot	Slot- oder Modul Nummer	BYTE

Tabelle 140: Ausgänge Hilfsfunktionsbaustein DEID

6.10.4 Hilfsfunktionsbaustein ID

(Identifikationsnummer generieren)

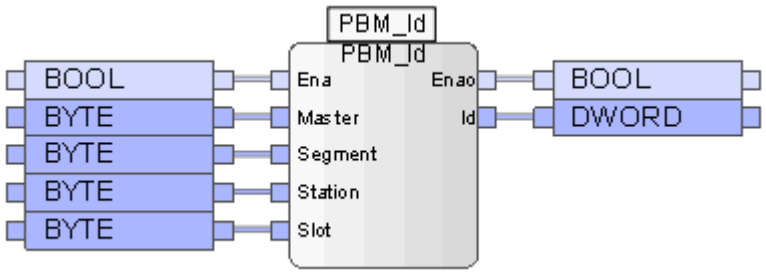


Bild 44: Hilfsfunktionsbaustein ID

Der Hilfsfunktionsbaustein **ID** generiert aus vier Bytes einen Identifier (Identifikationsnummer), die von anderen Hilfsfunktionsbausteinen genutzt wird.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
Ena	Nicht genutzt	BOOL
Master	Busadresse	BYTE
Segment	Segment	BYTE
Station	Busadresse des Slave	BYTE
Slot	Slot- oder Modul Nummer	BYTE

Tabelle 141: Eingänge Hilfsfunktionsbaustein ID

Ausgänge	Beschreibung	Typ
Enao	Nicht genutzt	BOOL
Id	Identifikationsnummer des Slaves	DWORD

Tabelle 142: Ausgänge Hilfsfunktionsbaustein ID

6.10.5 Hilfsfunktionsbaustein NSLOT

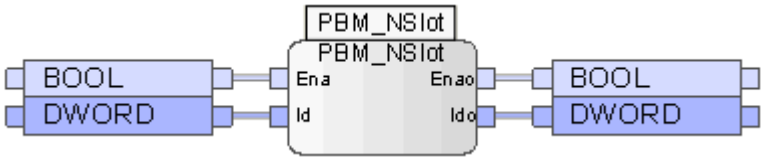


Bild 45: Hilfsfunktionsbaustein NSLOT

Der Hilfsfunktionsbaustein **NSLOT** generiert aus einem Identifier einen neuen Identifier, der den nächsten Slot im gleichen Slave adressiert. Ena muss auf TRUE gesetzt werden, damit der Hilfsfunktionsbaustein läuft.

Enao ist TRUE, wenn das Ergebnis am Ausgang Ido gültig ist.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
Ena	Solange TRUE anliegt, läuft der Baustein.	BOOL
Id	Identifikationsnummer des Slaves	DWORD

Tabelle 143: Eingänge Hilfsbaustein NSLOT

Ausgänge	Beschreibung	Typ
Enao	TRUE = Ergebnis gültig FALSE = Keine weiteren Slot-Nummern	BOOL
Ido	Identifikationsnummer des Slaves	DWORD

Tabelle 144: Ausgänge Hilfsbaustein NSLOT

6.10.6 Hilfsfunktionsbaustein SLOT

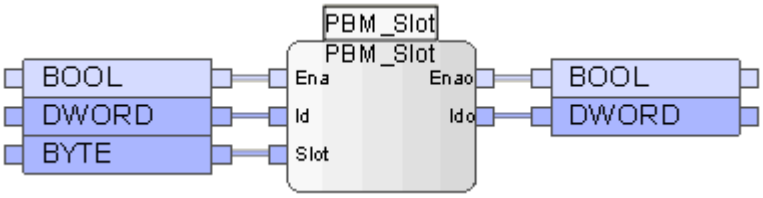


Bild 46: Hilfsfunktionsbaustein SLOT

Der Hilfsfunktionsbaustein **SLOT** generiert aus einem Identifier und einer Slot-Nummer einen neuen Identifier, der den gleichen Slave adressiert, wie der alte Identifier, jedoch mit der neuen Slot-Nummer.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
Ena	Nicht genutzt	BOOL
Id	Logische Adresse der Slave-Komponente (Slave-ID und Slot-Nummer)	DWORD
Slot	Neue Slot- oder Modul Nummer	BYTE

Tabelle 145: Eingänge Hilfsbaustein SLOT

Ausgänge	Beschreibung	Typ
Enao	Nicht genutzt	BOOL
Ido	Identifikationsnummer des Slaves	DWORD

Tabelle 146: Ausgänge Hilfsbaustein SLOT

6.10.7 Hilfsfunktionsbaustein STDDIAG

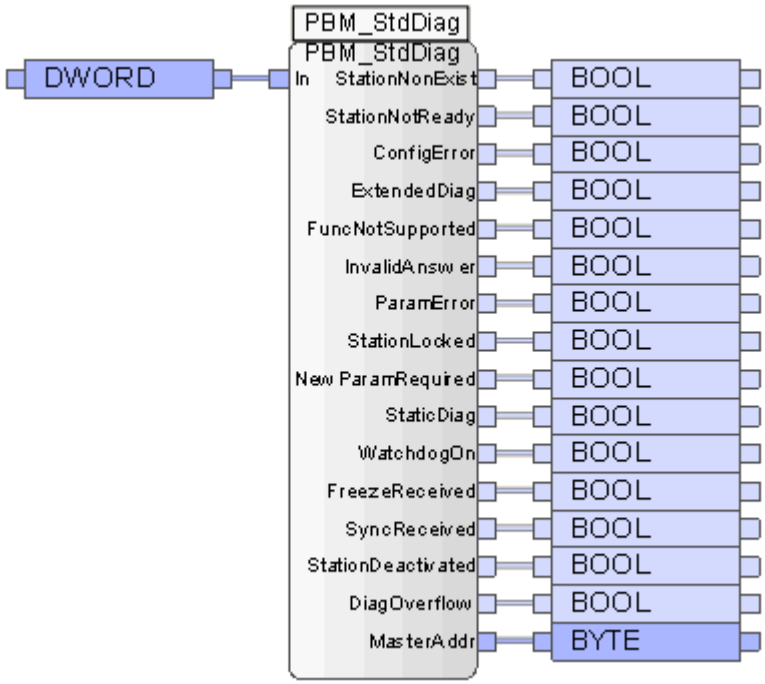


Bild 47: Hilfsfunktionsbaustein STDDIAG

Der Hilfsfunktionsbaustein Standard-Diagnose **STDDIAG** dekodiert die Standarddiagnose eines PROFIBUS DP Slaves.

Die Ausgänge vom Typ BOOL des Funktionsbaustein **STDDIAG** sind TRUE, wenn das dazugehörige Bit in der Standard-Diagnose gesetzt ist.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Eingänge	Beschreibung	Typ
IN	Standarddiagnose des Slave	DWORD

Tabelle 147: Eingänge Hilfsbaustein STDDIAG

Ausgänge	Beschreibung	Typ
StationNonExist	Slave existiert nicht	BOOL
StationNotReady	Slave nicht bereit	BOOL
ConfigError	Konfigurationsfehler	BOOL
ExtendedDiag	Erweiterte Diagnose folgt	BOOL
FuncNotSupported	Funktion nicht unterstützt	BOOL
InvalidAnswer	Ungültige Antwort vom Slave	BOOL
ParamError	Parametrierfehler	BOOL
StationLocked	Slave von anderem Master gesperrt	BOOL
NewParamRequired	Neue Parametrierdaten erforderlich	BOOL
StaticDiag	Statische Diagnose	BOOL
WatchdogOn	Watchdog aktiviert	BOOL
FreezeReceived	Freeze-Kommando erhalten	BOOL
SyncReceived	Sync-Kommando erhalten	BOOL
StationDeactivated	Slave wurde deaktiviert	BOOL
DiagOverflow	Diagnose überlauf	BOOL
MasterAddr	Busadresse des Masters	BYTE

Tabelle 148: Ausgänge Hilfsbaustein STDDIAG

So lesen Sie die Standard-Diagnose des PROFIBUS DP Slave aus:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, **PROFIBUS DP Master**.
2. Rechtsklicken auf **PROFIBUS Slave** und **Edit** wählen.
3. Globale Variable vom Typ DWORD auf das Feld *Standard-Diagnose* ziehen.
4. Diese Globale Variable mit dem Eingang des Funktionsbausteins **STDDIAG** verbinden.

6.11 Fehlercodes der Funktionsbausteine

Wenn ein Funktionsbaustein ein Kommando nicht korrekt ausführen konnte, wird am Ausgang **A_Status** ein Fehlercode ausgegeben. Die Bedeutung des Fehlercodes entnehmen Sie der folgenden Tabelle:

Fehlercode	Symbol	Beschreibung
16#40800800	TEMP_NOT_AVAIL	Dienst steht vorübergehend nicht zur Verfügung
16#40801000	INVALID_PARA	Ungültiger Parameter
16#40801100	WRONG_STATE	Slave unterstützt kein DP-V1
16#40808000	FATAL_ERR	Fataler Programmfehler
16#40808100	BAD_CONFIG	Konfigurationsfehler im Datenbereich
16#40808200	PLC_STOPPED	Steuerung wurde gestoppt
16#4080A000	READ_ERR	Fehler beim Lesen eines Records
16#4080A100	WRITE_ERR	Fehler beim Schreiben eines Records
16#4080A200	MODULE_FAILURE	Fehler nicht näher spezifizierbar
16#4080B000	INVALID_INDEX	Index ist ungültig
16#4080B100	WRITE_LENGTH	Falsche Länge beim Schreiben
16#4080B200	INVALID_SLOT	Slot-Nummer ist ungültig
16#4080B300	TYPE_CONFLICT	Falscher Typ
16#4080B400	INVALID_AREA	Falscher Lese- oder Schreibbereich
16#4080B500	STATE_CONFLICT	Master im falschen Zustand
16#4080B600	ACCESS_DENIED	Slave nicht aktiv (oder ähnliches)
16#4080B700	INVALID_RANGE	Falscher Lese- oder Schreibbereich
16#4080B800	INVALID_PARAMETER	Falscher Parameterwert
16#4080B900	INVALID_TYPE	Falscher Parametertyp
16#4080C300	NO_RESOURCE	Slave nicht vorhanden
16#4080BA00	BAD_VALUE	Ungültiger Wert
16#4080BB00	BUS_ERROR	Busfehler
16#4080BC00	INVALID_SLAVE	Ungültige Slave-Id
16#4080BD00	TIMEOUT	Timeout aufgetreten
16#4080C000	READ_CONSTRAIN	Lesebeschränkung
16#4080C100	WRITE_CONSTRAIN	Schreibbeschränkung
16#4080C200	BUSY	ein Baustein dieser Art ist bereits aktiv
16#4080C300	NO_RESOURCE	Slave nicht aktiv

Tabelle 149: Fehlercodes Funktionsbausteine

6.12 Control-Panel (PROFIBUS DP Master)

Im Control-Panel kann der Anwender die Einstellungen des PROFIBUS DP Master überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. Zykluszeit, Bus-Zustand usw.) des Masters und der zugehörigen Slaves angezeigt.

So öffnen Sie das Control Panel zur Überwachung des PROFIBUS DP Master:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **PROFIBUS DP Master** wählen.

6.12.1 Kontextmenü (PROFIBUS Master)

Aus dem Kontextmenü des selektierten PROFIBUS DP Master können die folgenden Kommandos gewählt werden:

Offline:

Schaltet den selektierten PROFIBUS DP Master aus. Ist der Master ausgeschaltet, erfolgen keine Aktivitäten.

Stop:

Stoppt den selektierten PROFIBUS DP Master. Der PROFIBUS DP Master nimmt weiterhin am Token-Protokoll teil, sendet aber keine Daten an die Slaves.

Clear:

Durch Betätigen der Schaltfläche CLEAR wird der selektierte PROFIBUS DP Master in einen sichereren Zustand gesetzt und tauscht nun sichere Daten mit den Slaves aus. Die Ausgangsdaten, die zu den Slaves gesendet werden, enthalten nur Nullen. FailSafe-Slaves erhalten FailSafe-Telegramme, die keine Daten enthalten. Die Eingangsdaten von den Slaves werden vom PROFIBUS DP Master ignoriert und stattdessen Initialwerte im Anwenderprogramm verwendet.

Operate:

Startet den selektierten PROFIBUS DP Master. Der PROFIBUS DP Master tauscht zyklisch E/A-Daten mit den Slaves aus.

Reset Statistik:

Die Schaltfläche Statistikwerte zurücksetzen setzt die statistischen Daten (Zykluszeit min, max usw.) auf null zurück.

6.12.2 Anzeigefeld (PROFIBUS Master)

In dem Anzeigefeld werden die folgenden Werte des selektierten PROFIBUS DP Master angezeigt.

Element	Beschreibung
Name	Name des PROFIBUS DP Master
Baudrate [bps]	Baudrate des Masters. Der Master kann mit allen Baudraten, die im Standard spezifiziert sind kommunizieren. Zykluszeiten sind bis zu einer Untergrenze von 2 ms möglich.
Feldbusschnittstelle	FB1, FB2
Feldbusadresse	Busadresse des Master (0 ... 125)
Master-Zustand	Zeigt den momentanen Protokollzustand an (siehe Kapitel 6.12.3). 0 = OFFLINE 1 = STOP 2 = CLEAR 3 = OPERATE 100 = UNDEFINED
Bus-Zustand	Fehlercode Busfehler 0 ... 6: 0 = OK 1 = Adressfehler: die Adresse des Masters ist auf dem Bus bereits vorhanden. 2 = Busstörung: es wurde eine Störung auf dem Bus registriert, z. B. Bus nicht richtig abgeschlossen, mehrere Teilnehmer senden gleichzeitig. 3 = Protokollfehler: ein fehlerhaft codiertes Paket wurde empfangen. 4 = Hardwarefehler: die Hardware hat einen Fehler gemeldet, z. B. bei zu knapp eingestellten Zeiten. 5 = Unbekannter Fehler: Master hat Zustand aus unbekanntem Grund gewechselt. 6 = Controller Reset: Bei schweren Busstörungen geht mitunter der Controller-Chip in einen undefinierten Zustand und wird zurückgesetzt. Der Fehlercode bleibt solange anliegen, bis der Busfehler behoben ist.
Anzahl BusFehler	Anzahl der Busfehler bisher
CPU-Last (projektiert) [%]	Anzeige der projektierten Auslastung des COM-Moduls für dieses Protokoll.
CPU-Last (tatsächliche) [%]	Tatsächliche Auslastung des COM-Moduls für dieses Protokoll.

Element	Beschreibung
MSI [ms]	Min. Slave Interval in ms, Auflösung 0.1ms
TTR [ms]	Target Rotation Time in ms, Auflösung 0.1ms
Letzte Zykluszeit [ms]	Letzte PROFIBUS DP Zykluszeit [ms]
Minimale Zykluszeit [ms]	Minimale PROFIBUS DP Zykluszeit [ms]
Mittlere Zykluszeit [ms]	Mittlere PROFIBUS DP Zykluszeit [ms]
Maximale Zykluszeit [ms]	Maximale PROFIBUS DP Zykluszeit [ms]

Tabelle 150: Anzeigenfeld PROFIBUS Master

6.12.3 Zustand des PROFIBUS DP Master

Der Zustand des Masters wird im Anzeigefeld des Control Panel angezeigt und kann mit der Statusvariable Master Verbindungszustand im Anwenderprogramm ausgewertet werden.

Master Zustand	Master Zustand
OFFLINE	Der Master ist ausgeschaltet und es erfolgen keine Busaktivitäten.
STOP	Der Master nimmt am Token-Protokoll teil, sendet aber keine Daten an die Slaves.
CLEAR	Der Master ist im sicheren Zustand und tauscht Daten mit den Slaves aus. ▪ Die Ausgangsdaten, die zu den Slaves gesendet werden, enthalten nur Nullen. ▪ Die FailSafe-Slaves erhalten FailSafe-Telegramme (diese enthalten keine Daten). ▪ Die Eingangsdaten von den Slaves werden ignoriert und stattdessen werden Initialwerte verwendet.
OPERATE	Der Master ist im Arbeitsmodus und tauscht zyklisch E/A-Daten mit den Slaves aus.
UNDEFINED	Firmware Update des PROFIBUS DP Master Moduls läuft gerade.

Tabelle 151: PROFIBUS DP Master Zustand

6.12.4 Verhalten des PROFIBUS DP Master

Verhalten des PROFIBUS DP Master in Abhängigkeit vom Betriebszustand der Steuerung.

Zustand der Steuerung	Verhalten des HIMA PROFIBUS DP Master
STOPP *)	Steuerung in STOPP, dann ist der Master im Zustand OFFLINE.
RUN	Steuerung in RUN, der Master versucht in den Zustand OPERATE zu gelangen.
STOPP	Steuerung in STOPP, der Master geht in den Zustand CLEAR. Ist der Master bereits in STOPP oder OFFLINE, bleibt er in diesem Zustand.

*) Nach Einschalten der Steuerung oder nach Laden der Konfiguration

Tabelle 152: PROFIBUS DP Master Verhalten

6.12.5 Funktion der FBx LED beim PROFIBUS DP Master

Der Zustand der seriellen PROFIBUS DP Kommunikation wird mit der FBx LED der jeweiligen konfigurierten seriellen Schnittstellen (fb1, fb2) angezeigt.

FBx LED	Beschreibung
AUS	Keine oder ungültige Konfiguration des PROFIBUS DP Master.
Blinkt im 2 Sekundentakt	Gültige Konfiguration. Der PROFIBUS DP Master befindet sich im Zustand OFFLINE oder STOPP.
AN	Der PROFIBUS DP Master befindet sich im Zustand OPERATE oder CLEAR und tauscht Daten mit allen aktivierten Slaves aus.
Blinkt im Sekundentakt	Mindestens ein Slave ist ausgefallen.

Tabelle 153: LED FBx (PROFIBUS DP Master)

6.13 HIMA PROFIBUS DP Slave

Dieses Kapitel beschreibt die Eigenschaften des HIMA PROFIBUS DP Slave und die Menüfunktionen in SILworX, die zur Konfiguration des HIMA PROFIBUS DP Slave benötigt werden.

Benötigte Ausstattung und Systemanforderungen

Element	Beschreibung
HIMA Steuerung	HIMax mit COM-Modul
COM-Modul	Das HIMax COM-Modul muss an der verwendeten seriellen Feldbusschnittstelle (FB1 oder FB2) mit einem optionalen HIMA PROFIBUS DP Slave Submodul ausgerüstet sein. Schnittstellenbelegung, siehe Kapitel 3.7.
Aktivierung	Freischaltung durch Aufsteckmodul, siehe Kapitel 3.5.

Tabelle 154: Ausstattung und Systemanforderungen HIMA PROFIBUS DP Slave

PROFIBUS DP Slave Eigenschaften

Element	Beschreibung
Typ des HIMA PROFIBUS DP Slaves	DP-V0
Übertragungsrate	9,6 kbit/s ... 12 Mbit/s
Busadresse	0 ... 125
Max. Anzahl Slaves	Es kann für jedes COM-Modul ein HIMA PROFIBUS DP Slave konfiguriert werden.
Prozessdatenmenge eines HIMA PROFIBUS DP Slaves	DP-Output: max. 192 Bytes DP-Input: max. 240 Bytes Insgesamt jedoch: max. 256 Bytes
Protokoll Watchdog	Ist die COM im Zustand RUN und die Verbindung zum PROFIBUS DP Master geht verloren, dann wird dies vom DP Slave nach Ablauf des Watchdog-Timeouts (muss im Master parametriert werden) erkannt. In diesem Fall werden die DP-Output-Daten (Eingangsdaten aus Sicht der Ressource) auf ihren Initialwert zurückgesetzt und das *Daten gültig*-Flag (Statusvariable des DP Slave-Protokolls) auf FALSE gesetzt.

Tabelle 155: Eigenschaften HIMA PROFIBUS DP Slave

6.13.1 Anlegen eines HIMA PROFIBUS DP Slave

So legen Sie einen neuen HIMA PROFIBUS DP Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **PROFIBUS DP Slave** wählen um einen neuen PROFIBUS DP Slave hinzuzufügen.
3. Im Kontextmenü vom PROFIBUS DP Slave **Edit** wählen.
4. Im Register **Eigenschaften** das **Modul** und die **Schnittstelle** auswählen.

6.14 Menüfunktionen PROFIBUS DP Slave

6.14.1 Edit

Das Dialogfenster **Edit** des PROFIBUS DP Master enthält die folgenden Register.

Prozessvariablen

Im Register **Prozessvariablen** werden die Variablen zum Empfangen und Senden angelegt.

Eingangsvariablen

Die Variablen, die von dieser Steuerung empfangen werden sollen, werden im Bereich *Eingangssignale* eingetragen.

Im Bereich *Eingangssignale* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Sendevariablen des Kommunikationspartners sein.

Ausgangsvariablen

Die Variablen für den zyklischen Datenaustausch, die von dieser Steuerung gesendet werden sollen, werden im Bereich *Ausgangssignale* eingetragen.

Im Bereich *Ausgangssignale* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Empfangsvariablen des Kommunikationspartners sein.

Systemvariablen

Im Register **Systemvariablen** wird festgelegt, welche Variablen in die Steuerung eingelesen werden sollen.

Das Register **Systemvariablen** stellt die folgenden Systemvariablen bereit, die es erlauben, den Zustand des PROFIBUS DP Slave im Anwenderprogramm auszuwerten.

Element	Beschreibung
Aktuelle Baudrate	Baudrate, mit der das PROFIBUS-DP-Slave-Protokoll aktuell arbeitet.
Daten gültig	Ist die Statusvariable *Daten gültig* auf TRUE gesetzt, dann hat der Slave gültige Import-Daten des Masters empfangen. Die Statusvariable wird auf FALSE gesetzt wenn die Watchdog-Zeit beim Slave abgelaufen ist. Standardwert: FALSE Hinweis: Wurde der Watchdog des Slaves durch den Master nicht aktiviert und die Verbindung geht verloren, so behält die Statusvariable *Daten gültig* den Wert TRUE, da der PROFIBUS DP Slave keine Möglichkeit hat, den Verbindungsverlust zu erkennen. Dieser Umstand ist bei der Verwendung dieser Variablen unbedingt zu beachten!
Fehlercode	Ist in dem PROFIBUS-DP-Slave-Protokoll ein Fehler aufgetreten, so wird dies in dieser Variablen übertragen. Es wird jeweils der aktuell aufgetretene Fehler angezeigt. Mögliche Werte (hexadezimal) sind: 0x00: kein Fehler 0xE1: falsche Parametrierung durch den PROFIBUS DP Master 0xD2: falsche Konfigurierung durch den PROFIBUS DP Master Standardwert: 0x00
Master-ID	Dies ist die ID des PROFIBUS DP Master, der den eigenen PROFIBUS DP Slave parametriert und konfiguriert hat. Mögliche Werte (dezimal) sind: 0-125: ID des Masters 255: Slave ist aktuell keinem Master zugeordnet Standardwert: 0xFF
Protokollzustand	Beschreibt den Zustand des PROFIBUS-DP-Slave-Protokolls. Mögliche Werte (hexadezimal) sind: 0xE1: Die Steuerung ist vom Bus getrennt bzw. nicht aktiv. 0xD2: Die Steuerung wartet auf eine Konfigurierung durch den PROFIBUS DP Master. 0xC3: Die Steuerung tauscht zyklisch Daten mit dem PROFIBUS DP Master aus. Standardwert: 0xE1
Slave-ID	Diese Variable enthält die PROFIBUS DP Slave-ID der Steuerung, mit der sie am Bus arbeitet. Diese Slave-ID wurde zuvor vom Anwender durch das PADT konfiguriert. Mögliche Werte (dezimal) sind: 0-125: PROFIBUS DP Slave-ID der eigenen Steuerung Standardwert: 0xFF
Watchdog-Zeit	Im Master parametrierte Watchdog-Zeit in Millisekunden. Siehe Kapitel 6.6.3.

Tabelle 156: Systemvariablen PROFIBUS DP Slave

6.14.2 Eigenschaften

Das Register **Eigenschaften** des HIMA PROFIBUS DP Slave enthält die folgenden Parameter zur Konfiguration des PROFIBUS DP Slave.

Die Vorgabewerte der Parameter *In einem Zyklus* und *Refresh Rate [ms]* sorgen für einen schnellen Datenaustausch der PROFIBUS DP Daten zwischen dem COM-Modul (COM) und der PROFIBUS DP Slave Hardware der HIMax Steuerung.

Diese Parameter sollten nur dann geändert werden, wenn eine Reduzierung der COM-Auslastung für eine Anwendung erforderlich ist und der Prozess dies zulässt.

i Die Änderung der Parameter wird nur dem erfahrenen Programmierer empfohlen.
Eine Erhöhung der Aktualisierungszeit der COM/ PROFIBUS DP Hardware bedeutet auch, dass die tatsächliche Aktualisierungszeit der PROFIBUS DP Daten erhöht wird. Die Zeitanforderungen der Anlage sind zu prüfen.

Beachten Sie auch den Parameter **Min. Slave Intervall [ms]** (siehe Kapitel 6.3.2 im Register Zeiten), der die minimale Aktualisierungszeit der PROFIBUS DP Daten zwischen PROFIBUS DP Master und PROFIBUS DP Slave festlegt.

Element	Beschreibung
Typ	PROFIBUS DP Slave
Name	Name für den PROFIBUS DP Slave.
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert
Modul	Auswahl des COM-Moduls, auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert : Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen. Deaktiviert: Kein Limit der CPU-Last für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%

<table>
<tr><td>Stationsadresse</td><td>Stationsadresse des Slaves.
Die Stationsadresse des Slaves darf auf dem Bus nur einmal vorhanden sein.
Wertebereich: 1 ... 125
Standardwert: 0</td></tr>
<tr><td>Refresh Rate [ms]</td><td>Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und der PROFIBUS DP Slave Hardware ausgetauscht werden.
Wertebereich: 4 ...1000
Standardwert: 10</td></tr>
<tr><td>Schnittstelle</td><td>Feldbusschnittstelle, die für den PROFIBUS DP Slave benutzt werden soll.
Wertebereich: fb1, fb2
Standardwert: keine</td></tr>
<tr><td>Baudrate [bps]</td><td>Baudrate mit welcher der Bus betrieben wird.
Mögliche Werte:
<table>
<tr><th>Wert</th><th>Baudrate</th><th>FB1</th><th>FB2</th></tr>
<tr><td>9600</td><td>9,6 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>19200</td><td>16,2 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>45450</td><td>45,45 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>93750</td><td>93,75 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>187500</td><td>187,5 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>500000</td><td>500 kbit/s</td><td>X</td><td>X</td></tr>
<tr><td>1500000</td><td>1,5 Mbit/s</td><td>X</td><td>X</td></tr>
<tr><td>3000000</td><td>3 Mbit/s</td><td>X</td><td>-</td></tr>
<tr><td>6000000</td><td>6 Mbit/s</td><td>X</td><td>-</td></tr>
<tr><td>12000000</td><td>12 Mbit/s</td><td>X</td><td>-</td></tr>
</table></td></tr>
</table>

Tabelle 157: Slave Eigenschaften: Register Allgemein

6.15 Control-Panel (PROFIBUS DP Slave)

Im Control-Panel kann der Anwender die Einstellungen des PROFIBUS DP Slave überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. Zykluszeit, Bus-Zustand usw.) des Slaves angezeigt.

So öffnen Sie das Control Panel zur Überwachung des PROFIBUS DP Slave:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **PROFIBUS DP Slave** wählen.

6.15.1 Kontextmenü (PROFIBUS DP Slave)

Aus dem Kontextmenü des selektierte PROFIBUS DP Slave können die folgenden Kommandos gewählt werden:

Aktivieren:

Aktiviert den selektierten Slave, der mit dem PROFIBUS DP Master nun Daten austauschen kann.

Deaktivieren:

Deaktiviert den selektierten Slave. Die Kommunikation wird beendet.

6.15.2 Anzeigefeld (PROFIBUS DP Slave)

In dem Anzeigefeld werden die folgenden Werte des selektierten PROFIBUS DP Master angezeigt.

Element	Beschreibung
Name	Name des PROFIBUS DP Slave
Feldbusschnittstelle	Zugeordnete Feldbusschnittstelle des Slave
Protokoll-Zustand	Verbindungszustand 0 = Deaktiviert, 1 = Inaktiv (versucht Verbindung aufzunehmen), 2 = Verbunden
Fehler-Zustand	Siehe Kapitel 6.14.1
Timeout	Im Master parametrierte Watchdog-Zeit in Millisekunden. Siehe Kapitel 6.6.3
Watchdog-Zeit [ms]	Wird im Master eingestellt. Siehe Kapitel 6.6.3.
Feldbusadresse	Siehe Kapitel 6.14.2.
Masteradresse	Adresse des PROFIBUS DP Master.
Baudrate [bps]	Aktuelle Baudrate. Siehe Kapitel 6.14.2.
CPU-Last (projektiert) [%]	Anzeige der projektierten Auslastung des COM-Moduls für dieses Protokoll.
CPU-Last (tatsächliche) [%]	Tatsächliche Auslastung des COM-Moduls für dieses Protokoll.

Tabelle 158: Anzeigenfeld (PROFIBUS DP Slave)

6.16 Funktion der FBx LED beim PROFIBUS DP Slave

Das COM-Modul signalisiert den Zustand des lokalen PROFIBUS-DP-Slave-Protokolls mittels einer der jeweiligen Feldbusschnittstelle zugeordneten LEDs. Die Zustände dieser LEDs sind in der folgenden Tabelle dargestellt:

FBx LED	Farbe	Beschreibung
AUS	Gelb	Das PROFIBUS-DP-Slave-Protokoll ist nicht aktiv! D.h. die Steuerung ist im Zustand STOPP oder es ist kein PROFIBUS DP Slave konfiguriert.
Blinkt im 2 Sekunden Takt	Gelb	Kein Datenaustausch! Der PROFIBUS DP Slave ist konfiguriert und bereit.
AN	Gelb	Das PROFIBUS-DP-Slave-Protokoll ist aktiv und befindet sich im Datenaustausch mit dem PROFIBUS DP Master.
AUS	Rot	PROFIBUS-DP-Slave-Protokoll ohne Störung.
Blinkt	Rot	Folgende Ereignisse führen zur Störung. Die Konfigurationen des PROFIBUS DP Masters und/oder des PROFIBUS DP Slaves sind fehlerhaft bzw. passen nicht zueinander. Überschreitung des Rechenzeitbugets Tritt länger als 5 Sekunden kein Fehlerereignis auf, so wechselt die Anzeige in den Zustand „Protokoll ohne Störung“.

Tabelle 159: LED FBx (PROFIBUS DP Slave)

7 Modbus

Die Modbus Kopplung der HIMax Systeme kann an nahezu alle Prozessleit- und Visualisierungssysteme entweder über die RS485-Schnittstellen oder über die Ethernet-Schnittstellen der Steuerungen erfolgen. HIMax Systeme können sowohl als Master, als auch als Slave betrieben werden.

Die Modbusfunktionalität erleichtert vor allem die Anbindung von Bedienpanels oder anderen Steuerungen. Durch die intensive Verbreitung und den weltweiten Projekteinsatz ist Modbus vielfach praxiserprobt.

Modbus Master (siehe Kapitel 7.1)
Die Redundanz des Modbus Masters müssen Sie im Anwenderprogramm so konfigurieren, dass dieses die redundanten Transportwege überwacht und die redundant übertragenen Prozessdaten dem jeweiligen Transportweg zuordnet.

Modbus Slave (siehe Kapitel 7.5.3)
Der Modbus Slave kann redundant konfiguriert werden.

i Bei Verwendung der Ethernet-Schnittstellen als Transportkanal müssen sich die HIMax Steuerung und der Kommunikationspartner im selben Subnet befinden oder bei Verwendung eines Routers die entsprechenden Routingeinträge besitzen.

7.1 HIMA Modbus Master

Die Datenübertragung zwischen dem HIMA Modbus Master und den Modbus Slaves können Sie über die serielle Schnittstelle (RS485) als auch über TCP/UDP (Ethernet) konfigurieren. Zudem können Sie den HIMA Modbus Master auch als Gateway (Modbus von TCP/UDP -> RS485) einsetzen.

Benötigte Ausstattung und Systemanforderungen

Element	Beschreibung
HIMA Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für Modbus TCP nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT D-SUB Anschlüsse FB1 und FB2 Wird Modbus RTU verwendet, muss das COM-Modul an den verwendeten seriellen Feldbusschnittstelle (FB1 oder FB2) mit einem optionalen HIMA RS485 Submodul ausgerüstet sein. Schnittstellenbelegung, siehe Kapitel 3.7.
Aktivierung	Jede der beiden Modbus Master Funktionen muss einzeln frei geschaltet werden, siehe Kapitel 3.5. Modbus Master RTU (RS 485), Modbus Master TCP

Tabelle 160: Ausstattung und Systemanforderung Modbus Master

Modbus Master Eigenschaften

Eigenschaft	Beschreibung
Modbus Master	Es kann für jedes COM-Modul ein Modbus Master konfiguriert werden. Ein Modbus Master kann gleichzeitig: TCP/UDP-Slaves, serielle Slaves auf mehreren seriellen Bussen bedienen und kann als Gateway von Modbus TCP auf Modbus RTU dienen.
Max. Anzahl Modbus Slaves	Ein Modbus Master kann bis zu 247 Slaves bedienen. 121 Modbus Slaves pro serielle Schnittstelle 64 TCP Slaves über TCP/IP-Verbindung 247 UDP Slaves über UDP/IP-Verbindung
Max. Anzahl Anforderungstelegramme	Es können bis zu 988 Anforderungstelegramme pro Modbus Master konfiguriert werden.
Max. Länge Anforderungstelegramm	Mit HIMA speziefischen Anforderungstelegrammen 1100 Byte, siehe Kapitel 7.5.2.
Max. Größe der Prozessdaten	Insgesamt können 128 kB Daten gesendet und 128 kB Daten empfangen werden. i Die Statusbytes des Masters und die Statusbytes von jedem zugeordneten Slave müssen von der max. Größe der Prozessdaten (128 kB) subtrahiert werden.
Darstellungsformat der Modbus-Daten	Die HIMax Steuerung verwenden das Big Endian Format. Beispiel 32 Bit Daten (z. B. DWORD, DINT):

32 Bit Daten (hex)	0x12345678			
Speicher-Offset	0	1	2	3
Big Endian (HIMax)	12	34	56	78
Middle Endian (H51q)	56	78	12	34
Little Endian	78	56	34	12

Tabelle 161: Eigenschaften Modbus Master

Nach der Norm sind insgesamt drei Repeater zulässig, sodass maximal 121 Slaves pro serielle Schnittstelle eines Masters möglich sind.

7.2 Modbus Beispiel

In diesem Beispiel tauscht ein HIMA Modbus Master Daten mit einem HIMA Modbus Slave über Modbus TCP aus. Die beiden Steuerungen werden über die Ethernet-Schnittstellen der Kommunikationsmodule verbunden.

i Befinden sich der Modbus Master und der Modbus Slave in verschiedenen Subnetzen, müssen in der Routing-Tabelle die entsprechenden benutzerdefinierten Routen eingetragen werden.

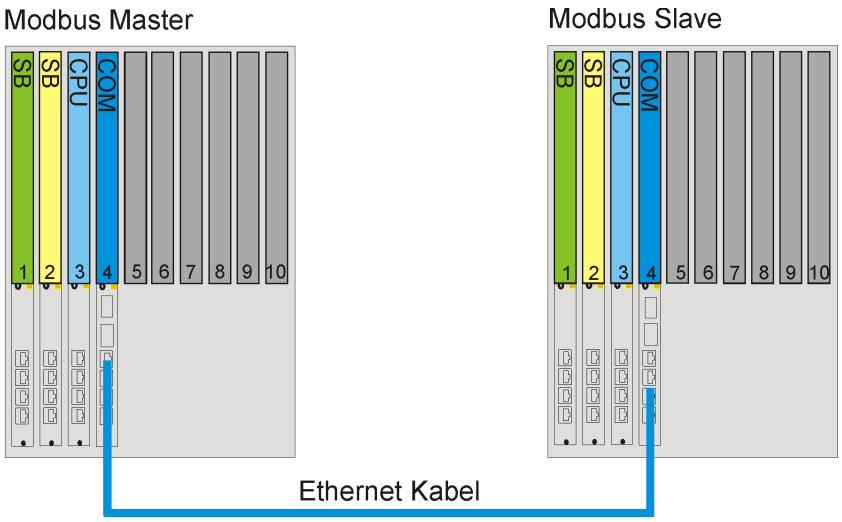


Bild 48: Kommunikation über Modbus TCP

Folgende Globale Variablen müssen Sie für dieses Beispiel in SILworX anlegen:

Globale Variablen	Typ
Master->Slave_BOOL_00	BOOL
Master->Slave_BOOL_01	BOOL
Master->Slave_BOOL_02	BOOL
Master->Slave_WORD_00	WORD
Master->Slave_WORD_01	WORD
Slave->Master_WORD_00	WORD
Slave->Master_WORD_01	WORD

7.2.1 Konfiguration des Modbus TCP Slave

So legen Sie einen neuen HIMA Modbus Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **Modbus-Slave-Set** wählen, um einen neuen Modbus-Slave-Set hinzuzufügen.
3. Im Kontextmenü des Modbus-Slave-Set **Edit** wählen und **Eigenschaften Modbus-Slave-Set** öffnen, Standardwerte beibehalten.
4. Register **Modbus-Slave** wählen und folgende Einstellungen vornehmen:
   - **COM Modul** wählen
   - **TCP aktivieren** aktivieren
   - Die restlichen Parameter behalten die Standardwerte.

So konfigurieren Sie die Bit Eingangsvariablen des Modbus Slave:

i Im Register Bitvariablen sind die boolschen Variablen einzutragen, die der Master Bitweise adressiert (Funktionscode 1, 2, 5, 15).

1. Im Kontextmenü des Modbus Slave **Edit**, **Bitvariablen** wählen.
2. In der **Objektauswahl** die folgenden Globalen Variablen auswählen und diese per Drag&Drop in den Bereich **Bit Eingänge** ziehen.

Bit Adresse	Bit Variable	Typ
0	Master->Slave_BOOL_00	BOOL
1	Master->Slave_BOOL_01	BOOL
2	Master->Slave_BOOL_02	BOOL

3. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Eingänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So konfigurieren Sie die Register Eingangsvariablen des Modbus Slave:

i Im Register Registervariablen sind die Variablen einzutragen, die der Master Register-weise adressiert (Funktionscode 3, 4, 6, 16 ,23).

1. Im Kontextmenü des Modbus Slave **Edit**, **Registervariablen** wählen.
2. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in den Bereich **Register Eingänge** ziehen.

Register Adresse	Register Variablen	Typ
0	Master->Slave_WORD_00	WORD
1	Master->Slave_WORD_01	WORD

3. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Eingänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So konfigurieren Sie die Register Ausgangsvariablen des Modbus Slave:

1. Im Kontextmenü des Modbus Slave **Edit**, **Registervariablen** wählen.
2. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in den Bereich **Register Ausgänge** ziehen.

Register Adresse	Register Variablen	Typ
0	Slave->Master_WORD_00	WORD
1	Slave->Master_WORD_01	WORD

3. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Ausgänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So Prüfen Sie die Modbus TCP Slave Konfiguration:

1. Kontextmenü des Modbus TCP Slave öffnen und **Verifikation** wählen.
2. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

7.2.2 Konfiguration des Modbus TCP Master

So legen Sie den HIMA Modbus Master an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **Modbus Master** wählen um einen neuen Modbus Master hinzuzufügen.
3. Im Kontextmenü vom Modbus Master **Eigenschaften**, **Allgemein** wählen.
4. **COM Modul** auswählen.
   Die restlichen Parameter behalten die Standardwerte.

So erstellen Sie im Modbus Master die Verbindung zu dem Modbus TCP Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master, Ethernet Slaves** öffnen.
2. Rechtsklick auf **Ethernet Slaves** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **TCP/UDP-Slave** wählen und mit **OK** bestätigen.
4. Konfiguration des TCP/UDP-Slave im Modbus Master:
   - **Edit** zum zuweisen der Systemvariablen wählen, siehe Kapitel 7.5.10.
   - **Eigenschaften** zum konfigurieren der Eigenschaften wählen, siehe Kapitel 7.5.11.
     In den Eigenschaften des Slaves die **IP Adresse** des TCP/UDP-Slaves eintragen.

Die restlichen Parameter behalten die Standardwerte.

So konfigurieren Sie das Anforderungstelegramm zum schreiben der Bit Ausgangsvariablen:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Write Multiple Coils (15)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Write Multiple Coils (15)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Schreibbereichs, 0** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Write Multiple Coils (15)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Ausgangsvariablen** ziehen.

Offset	Bit Variablen	Typ
0	Master->Slave_BOOL_00	BOOL
1	Master->Slave_BOOL_01	BOOL
2	Master->Slave_BOOL_02	BOOL

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So konfigurieren Sie das Anforderungstelegramm zum schreiben der Register Ausgangsvariablen:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Write Multiple Registers (16)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Write Multiple Registers (16)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Schreibbereichs, 0** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Write Multiple Registers (16)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Ausgangsvariablen** ziehen.

Offset	Register Variablen	Typ
0	Master->Slave_WORD_00	WORD
1	Master->Slave_WORD_01	WORD

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So definieren Sie im Modbus Master das Anforderungstelegramm zum lesen der Eingangsvariablen:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Read Holding Registers (03)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (03)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Lesebereichs, 0** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (03)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Eingangsvariablen** ziehen.

Offset	Register Variablen	Typ
0	Slave->Master_WORD_00	WORD
1	Slave->Master_WORD_01	WORD

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Eingangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So Prüfen Sie die Modbus TCP Master Konfiguration:

1. Kontextmenü des Modbus TCP Masters öffnen und **Verifikation** wählen.

So Prüfen Sie die Modbus TCP Master Konfiguration:

1. Kontextmenü des Modbus TCP Masters öffnen und **Verifikation** wählen.
2. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

So erstellen Sie den Code für die Steuerungen

1. Code Generator der Master und der Slave Ressource starten.
2. Sicherstellen, dass die Codes fehlerfrei generiert wurden.
3. Jeweilige Codes in die Master und Slave Steuerungen laden.

7.3 Beispiel zur Alternativen Register/Bit-Adressierung

In diesem Beispiel wird die Konfiguration aus dem Kapitel 7.2 um 16 boolsche Variablen im Registerbereich erweitert. Die 16 boolsche Variablen werden mit dem Anforderungstelegramm **Write Multiple Coils (15)** ausgelesen, siehe auch Kapitel 7.12.

So konfigurieren Sie die Eingangsvariablen im Modbus Slave:

1. Im Kontextmenü des Modbus Slave **Edit**, **Registervariablen** wählen.
2. In der **Objektauswahl** die 16 neuen boolsche Variablen auswählen und diese per Drag&Drop in den Bereich **Register Eingänge** ziehen.

Register Adresse	Register Variablen	Typ	
0	Master->Slave_WORD_00	WORD	
1	Master->Slave_WORD_01	WORD	
2	Master->Slave_BOOL_03 ..._18	BOOL	Neu hinzufügen

3. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Eingänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So konfigurieren Sie im Modbus Slave die Alternative Register/Bit-Adressierung:

1. Im Kontextmenü des Modbus Slave **Edit**, **Offsets** wählen und **Alternative Register/Bit-Adressierung verwenden** aktivieren.
2. Verwenden Sie für dieses Beispiel die folgenden Offsets für die alternativen Bereiche:

Register Bereich Offset Bits Input	1000
Register Bereich Offset Bits Output	1000
Bit Bereich Offset Register Input	8000
Bit Bereich Offset Register Output	8000

i Um mit dem Modbus Anforderungstelegramm **Write Multiple Coils (15)** auf boolsche Variablen im Bereich **Register Variablen** zuzugreifen, müssen die Variablen in den Bereich **Bitvariablen** gespiegelt sein.

So konfigurieren Sie im Modbus Master das Anforderungstelegramm zum schreiben der Ausgangsvariablen (BOOL):

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Write Multiple Coils (15)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Write Multiple Coils (15)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Schreibbereichs, 8032** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Write Multiple Coils (15)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Ausgangsvariablen** ziehen.

Offset	Gespiegelte Register Variable	Typ
0 bis 15	Master->Slave_BOOL_03..._18	BOOL

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

7.4 Menüfunktionen des HIMA Modbus Master

7.4.1 Edit

Das Dialogfenster **Edit** des Modbus Master enthält das folgende Register.

Systemvariablen

Das Register **Systemvariablen** stellt Systemvariablen bereit, die es erlauben, den Zustand des Modbus Master im Anwenderprogramm auszuwerten und den Modbus Master zu steuern.

Element	Beschreibung
Anzahl fehlerhafte Slave-Verbindungen	Anzahl der fehlerhaften Verbindungen mit Modbus Slaves, die im Zustand Activated sind. Deaktivierte Modbus Slaves werden hier nicht berücksichtigt.
Modbus-Master Aktivierungssteuerung	Hiermit kann der Modbus Master vom Anwenderprogramm gestoppt oder gestartet werden. 0:START 1:STOP
Modbus-Master Busfehler	Busfehler auf RS485, z. B. Telegrammfehler (unbekannte Codes etc), Längenfehler.
Modbus-Master Zustand	Der Modbus Master Zustand zeigt den momentanen Protokollzustand an: 1: OPERATE 0: OFFLINE
Reset aller Slave-Fehler	Mit einem Wechsel von FALSE->TRUE werden alle Slave-Fehler und Busfehler zurückgesetzt.

Tabelle 162: Systemvariablen Modbus Master

7.4.2 Eigenschaften

Die Menüfunktion **Eigenschaften** aus dem Kontextmenü des Modbus Master öffnet den Dialog *Eigenschaften*.

Der Dialog enthält die folgenden Register:

Allgemein

Im Register **Allgemein** werden der Name und die Beschreibung für den Modbus Master eingegeben. Zudem werden hier die Parameter eingestellt, wenn der Modbus Master zusätzlich als TCP und/oder UDP Gateway arbeiten soll.

Element	Beschreibung
Typ	Modbus Master
Name	Name für den Modbus Master
Beschreibung	Beschreibung für den Modbus Master
Modul	Auswahl des COM-Moduls, auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen. Deaktiviert: Kein Limit der CPU-Last für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
TCP-Gateway aktivieren	Ist das TCP Modbus Gateway aktiviert, muss mindestens eine Modbus -RS485 Schnittstelle konfiguriert werden.
TCP-Server-Port	Standard: 502 Es können auch andere TCP-Ports konfiguriert werden. Dabei ist die Port-Belegung bei der *Internet Corporation for Assigned Names and Numbers* (ICANN) zu beachten.
Maximale Anzahl TCP-Verbindungen als Server	Maximale Anzahl gleichzeitig offener TCP-Verbindungen als Server. Wertebereich:1 ... 64 Standardwert: 5
UDP Gateway aktivieren	Ist das UDP Modbus Gateway aktiviert, muss mindestens eine Modbus -RS485 Schnittstelle konfiguriert werden
UDP Port	Standard: 502 Es können auch andere UDP-Ports konfiguriert werden. Dabei ist die Port-Belegung bei der Internet Corporation for Assigned Names and Numbers (ICANN) zu beachten.
Maximale Länge der Queue	Länge der Gateway-Warteschlange für noch nicht beantwortete Anforderungstelegramme von anderen Mastern. Dies wird nur beachtet, wenn ein Gateway aktiviert ist. Wertebereich: 1 ... 20 Standardwert: 3

Tabelle 163: Allgemeine Eigenschaften Modbus Master

CPU/COM

Die Standardwerte für die Parameter sorgen für den schnellstmöglichen Datenaustausch der Modbus-Daten zwischen dem COM-Modul und dem CPU-Modul in der HIMax Steuerung.

Diese Parameter sollten nur dann geändert werden, wenn eine Reduzierung der COM- und/oder CPU-Auslastung für eine Anwendung erforderlich ist und der Prozess dies zulässt.

i Die Änderung der Parameter wird nur dem erfahrenen Programmierer empfohlen.

Eine Erhöhung der COM und CPU Aktualisierungszeit bedeutet auch, dass die tatsächliche Aktualisierungszeit der Modbus-Daten erhöht wird. Die Zeitanforderungen der Anlage sind zu prüfen.

Element	Beschreibung
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und CPU ausgetauscht werden. Ist die *Refresh Rate* Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 0 ... (2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert

Tabelle 164: Parameter COM/CPU

7.5 Modbus Funktionscodes (Anforderungstelegramme)

Mit den Modbus Funktionscodes (Anforderungstelegrammen) haben Sie die Möglichkeit, Variablen in beide Richtungen zu schreiben oder zu lesen. Es können einzelne Variablen oder mehrere aufeinander folgende Variablen gelesen oder geschrieben werden.

So erstellen Sie ein neues Anforderungstelegramm für einen TCP/UDP Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Ethernet-Slaves** einen **TCP/UDP Slave** wählen.
2. Rechtsklick auf **TCP/UDP Slave** und im Kontextmenü **Neu** wählen.
3. Aus dem Dialog **Neues Objekt** ein **Anforderungstelegramm** auswählen.

So erstellen Sie ein neues Anforderungstelegramm für einen Gateway-Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Modbus Gateway** einen **Gateway-Slave** wählen.
2. Rechtsklick auf **Gateway-Slave** und im Kontextmenü **Neu** wählen.
3. Aus dem Dialog **Neues Objekt** ein **Anforderungstelegramm** auswählen.

So erstellen Sie ein neues Anforderungstelegramm für einen RS485 Modbus Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Serieller Modbus** einen **Modbus Slave** wählen.
2. Rechtsklick auf **Modbus Slave** und im Kontextmenü **Neu** wählen.
3. Aus dem Dialog **Neues Objekt** ein **Anforderungstelegramm** auswählen.

7.5.1 Modbus Standard Funktionscodes

Folgende Modbus Standard Funktionscodes werden vom HIMA Modbus Master unterstützt.

Element	Code	Typ	Bedeutung
READ COILS	01	BOOL	Lesen mehrerer Variablen (BOOL) aus dem Slave.
READ DISCRETE INPUTS	02	BOOL	Lesen mehrerer Variablen (BOOL) aus dem Slave.
READ HOLDING REGISTERS	03	WORD	Lesen mehrerer Variablen beliebigen Typs aus dem Slave.
READ INPUT REGISTERS	04	WORD	Lesen mehrerer Variablen beliebigen Typs aus dem Slave.
WRITE SINGLE COIL	05	BOOL	Schreiben eines einzelnen Signals (BOOL) in den Slave.
WRITE SINGLE REGISTER	06	WORD	Schreiben eines einzelnen Signals (WORD) in den Slave.
WRITE MULTIPLE COILS	15	BOOL	Schreiben mehrerer Variablen (BOOL) in den Slave.
WRITE MULTIPLE REGISTERS	16	WORD	Schreiben mehrerer Variablen beliebigen Typs in den Slave.
READ WRITE HOLDING REGISTERS	23	WORD	Schreiben und Lesen mehrerer Variablen beliebigen Typs in und aus dem Slave.

Tabelle 165: Modbus-Funktionscodes

i Weitere Informationen zu Modbus finden Sie in der Spezifikation *Modbus Application Protocol Specification* www.modbus.org

7.5.2 HIMA spezifische Funktionscodes

Die HIMA speziefischen Funktionscodes entsprechen den Standard Modbus Funktionscodes. Die zwei Unterschiede sind die maximal zulässige Prozessdatenlänge von 1100 Bytes und das Format von Request und Response-Header.

Element	Code	Typ	Bedeutung
Read Coils Extended	100 (0x64)	BOOL	Entspricht dem Functioncode 01 Lesen mehrerer Variablen (BOOL) aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Discrete Inputs Extended	101 (0x65)	BOOL	Entspricht dem Functioncode 02 Lesen mehrerer Variablen (BOOL) aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Holding Registers Extended	102 (0x66)	WORD	Entspricht dem Functioncode 03 Lesen mehrerer Variablen beliebigen Typs aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Input Registers Extended	103 (0x67)	WORD	Entspricht dem Functioncode 04 Lesen mehrerer Variablen beliebigen Typs aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Write Multiple Coils Extended	104 (0x68)	BOOL	Entspricht dem Functioncode 15 Schreiben mehrerer Variablen (BOOL) in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Write Multiple Registers Extended	105 (0x69)	WORD	Entspricht dem Functioncode 16 Schreiben mehrerer Variablen beliebigen Typs in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Entspricht dem Functioncode 23 Schreiben und Lesen mehrerer Variablen beliebigen Typs in und aus dem Import-Bereich oder Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes (Anforderungstelegramm vom Modbus Master) 1100 Bytes (Antwort an den Master).

Format der Request und Response Header

Die Request und Response Header der HIMA speziefischen Modbus-Funktionscodes sind wie folgt aufgebaut:

Code	Request	Response
100 (0x64)	1 byte Functionscode 0x64 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260)	1 byte Functionscode 0x64 2 bytes Anzahl von Bytes = N N bytes Coil-Daten (8 Coils werden in ein Byte gepackt)
101 (0x65)	1 byte Functionscode 0x65 2 bytes Startadresse 2 bytes Anzahl von Discrete Inputs 1 ... 8800(0x2260)	1 byte Functionscode 0x65 2 bytes Anzahl von Bytes = N N bytes Discrete Inputs -Daten (8 Discrete Inputs werden in ein Byte gepackt)
102 (0x66)	1 byte Functionscode 0x66 2 bytes Startadresse 2 bytes Anzahl von Register 1 ... 550(0x226)	1 byte Functionscode 0x66 2 bytes Anzahl von Bytes = N N bytes Register-Daten
103 (0x67)	1 bytes Functionscode 0x67 2 bytes Startadresse 2 bytes Anzahl von Register 1 ... 550(0x226)	1 byte Functionscode 0x67 2 bytes Anzahl von Bytes = N N bytes Register-Daten
104 (0x68)	1 byte Functionscode 0x68 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260) 2 bytes Anzahl von Bytes = N N bytes Coil-Daten	1 byte Functionscode 0x68 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260)
105 (0x69)	1 byte Functionscode 0x69 2 bytes Startadresse 2 bytes Anzahl von Registern 1 ... 550(0x226) 2 bytes Anzahl von Bytes = N N bytes Register-Daten	1 byte Functionscode 0x69 2 bytes Startadresse 2 bytes Anzahl von Registern 1 ... 550(0x226)
106 (0x6A)	1 byte Functionscode 0x6a 2 bytes Lese-Startadresse 2 bytes Anzahl von Leseregistern 1 ... 550(0x226) 2 bytes Schreib-Startadresse 2 bytes Anzahl von Schreibregistern 1 ... 550(0x226) 2 bytes Anzahl von Bytes zum Schreiben=N N bytes Register-Daten	1 byte Functionscode 0x6a 2 bytes Anzahl von Bytes = N N bytes Register-Daten

7.5.3 Anforderungstelegramme zum Lesen

Mit den Read-Funktionscodes können Variablen aus dem Slave gelesen werden.

Ein Anforderungstelegramm des Modbus Master enthält neben der Modbus Funktion die Startadresse des Lese-/Schreibbereichs.

Zum Lesen von Variablen sendet der Modbus Master ein *Anforderungstelegramm zum Lesen* an den Modbus Slave.

Der Modbus Slave sendet daraufhin ein Antworttelegramm mit den angeforderten Variablen an den Modbus Master zurück.

So konfigurieren Sie ein Anforderungstelegramm zum Lesen:

1. Im Strukturbaum **Anforderungstelegramm** zum konfigurieren auswählen.
2. Rechtsklick auf **Anforderungstelegramm** und im Kontextmenü **Edit** wählen.
3. In der **Objektauswahl** eine Globale Variable wählen, die als Modbus Empfangsvariablen dienen soll und diese per Drag&Drop auf eine leere Stelle im Bereich **Eingangssignale** ziehen.
4. Diesen Schritt für jede weitere Modbus Empfangsvariable wiederholen.
5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

Die folgenden *Anforderungstelegramme zum Lesen* stehen zur Verfügung:

Read Coils (01) und Extended (100)

Lesen mehrerer Variablen (BOOL) aus dem Slave.

Element	Bedeutung
Typ	Modbus -Funktion Read Coils
Name	Beliebiger, eindeutiger Name, für die Modbus -Funktion
Beschreibung	Beschreibung für die Modbus -Funktion
Startadresse des Lesebereichs	0 ... 65535

Tabelle 166: Anforderungstelegramm Read Coils

Read Discrete Inputs (02) und Extended (101)

Lesen mehrerer Variablen (BOOL) aus dem Slave.

Element	Bedeutung
Typ	Modbus -Funktion Read Discrete Inputs
Name	Beliebiger, eindeutiger Name, für die Modbus -Funktion
Beschreibung	Beschreibung für die Modbus -Funktion
Startadresse des Lesebereichs	0 ... 65535

Tabelle 167: Anforderungstelegramm Read Discrete Inputs

Read Holding Registers (03) und Extended (102)

Lesen mehrerer Variablen beliebigen Typs aus dem Slave.

Element	Bedeutung
Typ	Modbus -Funktion Read Holding Registers
Name	Beliebiger, eindeutiger Name, für die Modbus -Funktion
Beschreibung	Beschreibung für die Modbus -Funktion
Startadresse des Lesebereichs	0 ... 65535

Tabelle 168: Anforderungstelegramm Read Holding Registers

Read Input Registers (04) und Extended (103)

Lesen mehrerer Variablen beliebigen Typs aus dem Slave

Element	Bedeutung
Typ	Modbus Funktion Read Input Registers
Name	Beliebiger, eindeutiger Name, für die Modbus Funktion
Beschreibung	Beschreibung für die Modbus Funktion
Startadresse des Lesebereichs	0 ... 65535

Tabelle 169: Anforderungstelegramm Read Input Registers

Anforderungstelegramm zum Lesen und Schreiben

Zum Lesen und Schreiben von Variablen sendet der Modbus Master ein *Anforderungstelegramm zum Lesen und Schreiben* an den Modbus Slave.

Zuerst schreibt der Modbus Master die definierten Schreibvariablen in den definierten Import-Bereich des Modbus Slave.

Anschließend liest der Modbus Master die definierten Lesevariablen aus dem definierten Export-Bereich des Modbus Slave.

i Die Funktionen Schreiben und Lesen sind auch bei dem *Anforderungstelegramme zum Lesen und Schreiben* voneinander unabhängig, sie werden nur in einem gemeinsamen Anforderungstelegramm gesendet.

Eine häufige Anwendung für das *Anforderungstelegramme zum Lesen und Schreiben* ist jedoch, dass die geschriebenen Variablen des Modbus Master wieder zurückgelesen werden. Damit wird überprüft, ob die gesendeten Variablen korrekt geschriebenen wurden.

So konfigurieren Sie ein Anforderungstelegramm zum Lesen und Schreiben:

1. Im Strukturbaum **Anforderungstelegramm** zum konfigurieren auswählen.
2. Rechtsklick auf **Anforderungstelegramm** und im Kontextmenü **Edit** wählen.

So konfigurieren Sie die Variablen zum Lesen:

1. Wählen Sie in der **Objektauswahl** eine Globale Variable, die Sie mit der neuen Modbus Empfangsvariablen verbinden wollen und ziehen Sie diese per Drag&Drop in die Spalte **Globale Variable** der Modbus Empfangsvariablen.
2. Diesen Schritt für jede weitere Modbus Empfangsvariable wiederholen.
3. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Eingangssignale** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

So konfigurieren Sie die Variablen zum Schreiben:

1. Wählen Sie in der **Objektauswahl** eine Globale Variable, die Sie mit der neuen Modbus Sendevariablen verbinden wollen und ziehen Sie diese per Drag&Drop in die Spalte **Globale Variable** der Modbus Sendevariablen.
2. Diesen Schritt für jede weitere Modbus Sendevariable wiederholen.
3. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Ausgangssignale** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

Read Write Holding Register (23) und Extended (106)

Schreiben und Lesen mehrerer Variablen beliebigen Typs in und aus dem Import-Bereich des Slaves.

Element	Bedeutung
Typ	Modbus-Funktion *Read Write Holding Registers*
Name	Beliebiger, eindeutiger Name, für die Modbus-Funktion
Beschreibung	Beschreibung für die Modbus-Funktion
Startadresse des Lesebereichs	0 ... 65535
Startadresse des Schreibbereichs	0 ... 65535

Tabelle 170: Register Read Write Holding

7.5.4 Anforderungstelegramm zum Schreiben

Mit den Write-Funktionscodes werden Variablen nur in den Importbereich eines Slaves geschrieben.

Ein Anforderungstelegramm des Modbus Master enthält neben der Modbus-Funktion die Startadresse des Lese-/Schreibbereichs.

Zum Schreiben von Variablen sendet der Modbus Master ein *Anforderungstelegramm zum Schreiben* an den Modbus Slave.

Der Modbus Slave schreibt die empfangenen Variablen in seinen Import-Bereich.

Im Dialog *Variablen zuweisen* eines *Anforderungstelegramms zum Schreiben* müssen die Variablen eingefügt werden, die der Modbus Master zum Modbus Slave schreibt.

So konfigurieren Sie ein Anforderungstelegramm zum Schreiben:

1. Im Strukturbaum **Anforderungstelegramm** zum konfigurieren auswählen.
2. Rechtsklick auf **Anforderungstelegramm** und im Kontextmenü **Edit** wählen.
3. In der **Objektauswahl** eine Globale Variable wählen, die als Modbus Sendevariable dienen soll und diese per Drag&Drop auf eine leere Stelle im Bereich **Sendesignale** ziehen.
4. Diesen Schritt für jede weitere Modbus Sendevariable wiederholen.
5. Kontextmenü durch einen Rechtsklick auf eine leere Stelle im Bereich **Sendesignale** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

Die folgenden *Anforderungstelegramme zum Schreiben* stehen zur Verfügung:

Write Multiple Coils (15) und Extended (104)

Schreiben mehrerer Variablen (BOOL) in den Import-Bereich des Slaves.

Element	Bedeutung
Typ	Modbus Funktion *Write Multiple Coils*
Name	Beliebiger, eindeutiger Name, für die Modbus Funktion
Beschreibung	Beschreibung für die Modbus Funktion
Startadresse des Schreibbereichs	0 ... 65535

Tabelle 171: Anforderungstelegramm Write Multiple Coils

Write Multiple Registers (16) und Extended (105)

Schreiben mehrerer Variablen beliebigen Typs in den Import-Bereich des Slaves.

Element	Bedeutung
Typ	Modbus Funktion *Write Multiple Registers*
Name	Beliebiger, eindeutiger Name, für die Modbus Funktion
Beschreibung	Beschreibung für die Modbus Funktion
Startadresse des Schreibbereichs	0 ... 65535

Tabelle 172: Anforderungstelegramm Write Multiple Registers

Write Single Coil (05)

Schreiben einer einzelnen Variablen (BOOL) in den Import-Bereich des Slaves.

Element	Bedeutung
Typ	Modbus Funktion *Write Single Coil*
Name	Beliebiger, eindeutiger Name, für die Modbus Funktion
Beschreibung	Beschreibung für die Modbus Funktion
Startadresse des Schreibbereichs	0 ... 65535

Tabelle 173: Anforderungstelegramm Write Single Coil (05)

Write Single Register (06)

Schreiben einer einzelnen Variablen (WORD) in den Import-Bereich des Slaves.

Element	Bedeutung
Typ	Modbus Funktion *Write Single Register*
Name	Beliebiger, eindeutiger Name, für die Modbus Funktion
Beschreibung	Beschreibung für die Modbus Funktion
Startadresse des Schreibbereichs	0 ... 65535

Tabelle 174: Anforderungstelegramm Write Single Register

7.5.5 Ethernet Slaves (TCP/UDP-Slaves)

Der Modbus Master kann mit bis zu 64 TCP/IP und 247 UDP/IP Slaves kommunizieren.

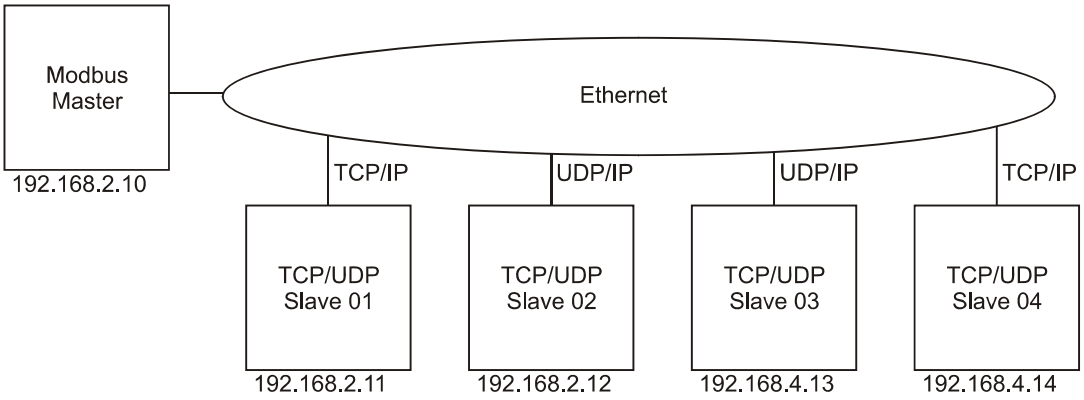


Bild 49: Modbus-Netzwerk

So erstellen Sie im Modbus Master eine neue Verbindung zu einem TCP/UDP Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Ethernet-Slaves** öffnen.
2. Rechtsklick auf **Ethernet-Slaves** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **TCP/UDP-Slaves** wählen und mit **OK** bestätigen.
4. Konfiguration des TCP/UDP-Slave im Modbus Master::
   Edit zum Zuweisen der Systemvariablen, siehe Kapitel 7.5.6.
   Eigenschaften zum Konfigurieren der Eigenschaften, siehe Kapitel 7.5.7.

i Befinden sich die TCP/UDP-Slaves und der Modbus Master in verschiedenen Subnetzen, müssen in der Routing-Tabelle die entsprechenden benutzerdefinierten Routen eingetragen werden.

7.5.6 Systemvariablen der TCP/UDP-Slaves

Das Register *Systemvariablen* stellt Systemvariablen bereit, die es erlauben, den Zustand des TCP/UDP Slave im Anwenderprogramm auszuwerten und zu steuern.

Der Status des TCP/UDP Slave kann im Anwenderprogramm mit den folgenden Statusvariablen ausgewertet werden:

Element	Beschreibung
Modbus-Slave Aktivierungssteuerung	Hiermit kann der TCP/UDP Slave vom Anwenderprogramm deaktiviert oder aktiviert werden. 0: Aktivieren 1: Deaktivieren
Modbus-Slave Fehler	Fehlercode Die Fehlercodes 0x01 ... 0x0b entsprechen den Exception Codes der Modbus-Protokollspezifikation. 0x00: Kein Fehler Exception Codes: 0x01: Ungültiger Funktionscode 0x02: Ungültige Adressierung 0x03: Ungültige Daten 0x04: (nicht verwendet) 0x05: (nicht verwendet) 0x06: Device Busy (nur Gateway) 0x08: (nicht verwendet) 0x0a: (nicht verwendet) 0x0b: No Response from Slave (nur Gateway) HIMA spezifische Codes: 0x10: Defekter Frame empfangen 0x11: Frame mit falscher Transaktions ID empfangen 0x12: Unerwartete Antwort empfangen 0x13: Antwort über falsche Verbindung erhalten 0x14: Falsche Antwort auf einen Schreib Auftrag 0xff: Slave Timeout
Modbus-Slave Zustand	Verbindungsstatus des TCP/UDP Slave: 0: Deaktiviert 1: Nicht verbunden 2: Verbunden

Tabelle 175: Systemvariablen TCP/UDP-Slaves

7.5.7 Eigenschaften TCP/UDP-Slaves

Zur Konfiguration der Verbindung zum TCP/UDP Slave müssen im Modbus Master die folgenden Parameter eingestellt werden.

Element	Beschreibung
Typ	TCP/UDP Slave
Name	Beliebiger eindeutiger Name für den TCP/UDP Slave
Beschreibung	Beliebige eindeutige Beschreibung für den TCP/UDP Slave
IP-Adresse	IP-Adresse des TCP/UDP Slave
Port	Standard: 502 Es können auch andere TCP/UDP-Ports konfiguriert werden. Dabei ist die Port-Belegung bei der Internet Corporation for Assigned Names and Numbers (ICANN) zu beachten.

Kommunikationsart IP-Protokoll	TCP oder UDP Standardwert: TCP
TCP-Verbindung nur bei Bedarf	Wenn das Transportprotokoll TCP ist wird hier eingestellt, ob die Verbindung zu diesem Slave nach jedem Datenaustausch automatisch abgebaut werden soll. TRUE: Die Verbindung abbauen. FALSE: Die Verbindung nicht abbauen. Standardwert: FALSE
Master-Slave Datenaustausch [ms]	Intervall für den Datenaustausch mit diesem Slave 1 bis (2^{31}-1). Konnte der Slave nach *Maximale Anzahl Sendewiederholungen* nicht erreicht werden, wird das Intervall *Master-Slave Datenaustausch* um das Vierfache hochgesetzt.
Maximale Anzahl Sendewiederholungen	Maximale Anzahl an Sendewiederholungen, falls Slave nicht antwortet. Die Anzahl der Sendewiederholungen kann beliebig eingestellt werden (0 ... 65535). Bei TCP/IP immer null, nicht änderbar. Empfohlen wird eine Anzahl von null bis acht Sendewiederholungen.
Receive Timeout [ms]	Receive Timeout für diesen Slave [ms]. Nach dieser Zeit wird ein neuer Sendeversuch gestartet.

Tabelle 176: Konfigurationsparameter

7.5.8 Modbus Gateway (TCP/UDP Gateway)

Der Modbus Master kann als Modbus Gateway arbeiten. In diesem Modus werden Master Requests, die das Gateway über Ethernet empfängt, an die an dem Gateway angeschlossenen RS485- und/oder Ethernet-Slaves weitergeleitet. Entsprechend werden die Antworten der Slaves über den Gatway an den Modbus Master weitergeleitet.

Bis zu 121 serielle Modbus Slaves können über die serielle Schnittstelle adressiert werden.

Der Bereich der Slave-Adressen ist 1 bis 247. Der Modbus Master 2 (Modbus Gateway) benötigt eine Modbus Master Lizenz, auch wenn nur der Modbus Gateway genutzt wird.

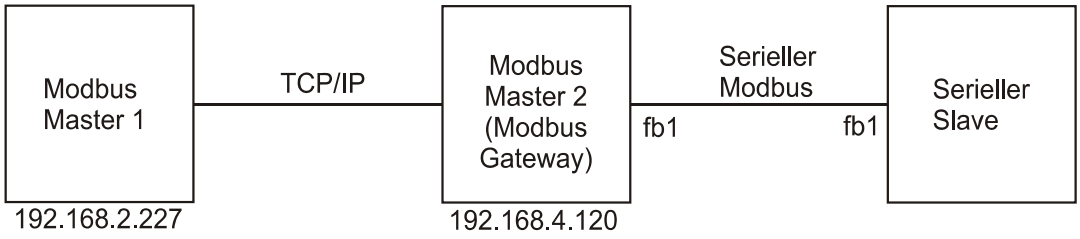


Bild 50: Modbus Gateway

i Befinden sich der Modbus Gateway und der Modbus Master in verschiedenen Subnetzen, müssen in der Routing-Tabelle die entsprechenden benutzerdefinierten Routen eingetragen werden.

Modbus Master 1:

So erstellen Sie im Modbus Master 1 die Verbindung zu dem Modbus Gateway:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master** öffnen.
2. Rechtsklick auf **Modbus Master** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **Modbus Gateway** wählen und mit **OK** bestätigen.
4. Konfiguration des Modbus Gateway im Modbus Master 1:
   Eigenschaften zum Konfigurieren der Eigenschaften, siehe Kapitel 7.5.15.
   In den Eigenschaften die **IP-Adresse** des Modbus Master 2 (Modbus Gateway) eintragen.

So erstellen Sie im Modbus Master 1 die Verbindung zu dem Gateway-Slave:

Im Modbus Master 1 muss der serielle Slave als Gateway-Slave angelegt werden.

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Modbus Gateway** öffnen.
2. Rechtsklick auf **Modbus Gateway** und Kontextmenü **Neu** wählen.
3. Aus der Liste **Gateway-Slave** wählen und mit **OK** bestätigen.
4. Konfiguration des Gateway-Slave im Modbus Master 1:
   Edit zum Zuweisen der Systemvariablen, siehe Kapitel 7.5.10.
   Eigenschaften zum Konfigurieren der Eigenschaften, siehe Kapitel 7.5.11.
   In den Eigenschaften des Slaves die **Serielle Adresse** des Gateway-Slaves eintragen.

So definieren Sie im Modbus Master 1 die Ein- und Ausgangsvariablen zu dem seriellen-Slave:

1. Rechtsklick auf **Gateway-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste die benötigten **Anforderungstelegramme** wählen.
3. Rechtsklick auf das jeweilige **Anforderungstelegramm** und im Kontextmenü **Edit** wählen. In Register *Prozessvariablen* die Ein- oder Ausgangsvariablen eintragen.

Modbus Master 2 (Modbus Gateway):

In den Eigenschaften des Modbus Master 02 muss die Gateway Funktion aktiviert werden. Damit werden die im Master 01 konfigurierten Gateway-Slaves mit den seriellen Slaves verbunden.

So aktiveren Sie im Modbus Master 2 die Gateway-Funktion:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master** öffnen.
2. Rechtsklick auf **Modbus Master** und im Kontextmenü **Eigenschaften** wählen.
3. Parameter **TCP-Gateway aktivieren** einschalten damit der Modbus Master zusätzlich als TCP-Gateway arbeitet.
4. Parameter **UDP Gateway aktivieren** einschalten, damit der Modbus Master zusätzlich als UDP Gateway arbeitet.

So konfigurieren Sie im Modbus Master 2 den seriellen Modbus:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master** öffnen.
2. Rechtsklick auf **Modbus Master** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **Serieller Modbus** wählen und mit **OK** bestätigen.
4. Konfiguration des **seriellen Modbus**
   Eigenschaften wählen und Schnittstelle, Baudrate etc. eintragen.

So konfigurieren Sie im Modbus Master 2 die Verbindung zu dem seriellen Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Serieller Modbus** öffnen.
2. Rechtsklick auf **serieller Modbus** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **Modbus Slave** wählen und mit **OK** bestätigen.
4. Konfiguration des **Modbus Slave**
   Eigenschaften wählen und **Slave-Adresse** des seriellen Slaves eintragen.

Serieller Slave

So konfigurieren Sie den seriellen Modbus Slave:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Slave** öffnen.
2. Rechtsklick auf **Modbus Slave** und im Kontextmenü **Edit** wählen.
3. Konfiguration des **Modbus Slave**
   Eigenschaften und **Slave-Adresse** des seriellen Slaves eintragen.

7.5.9 Eigenschaften Gateway

Der Modbus Master kommuniziert mit seinen Modbus Slaves über den Modbus Gateway.

Zur Konfiguration der Verbindung zum Modbus Gateway müssen im Modbus Master die folgenden Parameter eingestellt werden.

Element	Beschreibung
Typ	Modbus Gateway
Name	Beliebiger, eindeutiger Name, für den Gateway
Beschreibung	Beliebige, eindeutige Beschreibung für den TCP/UDP Slave
Kommunikations IP-Protokoll	TCP oder UDP Standardwert: TCP
IP-Adresse	IP-Adresse des Gateways, über welches der Modbus Master mit seinen Modbus Slave kommunizieren soll. Standardwert: (0.0.0.0)
Port	Standardwert: 502

Tabelle 177: Verbindungsparameter Modbus Gateway

7.5.10 Systemvariablen Gateway-Slave

Im Editor **Edit** stehen die drei Statusvariablen bereit:

Element	Beschreibung
Modbus-Slave Aktivierungssteuerung	Hiermit kann der Gateway-Slave vom Anwenderprogramm deaktiviert oder aktiviert werden. 0:Aktivieren 1:Deaktivieren
Modbus-Slave Fehler	Parameter wie bei TCP/UDP Slave, siehe Kapitel 7.5.6.
Modbus-Slave Zustand	Verbindungsstatus des Gateway-Slave: 0: Deaktiviert 1: Nicht verbunden 2: Verbunden

Tabelle 178: Statusvariablen Gateway-Slave

7.5.11 Eigenschaften Gateway-Slave

Zur Konfiguration der Verbindung zum Gateway-Slave müssen im Modbus Master die folgenden Parameter eingestellt werden.

Element	Beschreibung
Typ	Gateway-Slave
Name	Beliebiger eindeutiger Name für den Gateway-Slave
Beschreibung	Beliebige eindeutige Beschreibung für den Gateway-Slave
Slave-Adresse	1 ... 247
Restliche Parameter wie bei TCP/UDP Slave, siehe Kapitel 7.5.7	

Tabelle 179: Verbindungsparameter Gateway-Slave

7.5.12 Serieller Modbus

Der Modbus Master kann mit bis zu 247 seriellen Slaves kommunizieren. Nach der Norm sind insgesamt drei Repeater zulässig, sodass maximal 121 Busteilnehmer pro serielle Schnittstelle eines Masters möglich sind.

i Die Pin-Belegung der D-SUB-Anschlüsse (fb1, fb2) des HIMax COM-Moduls, siehe Kapitel 3.7.1.

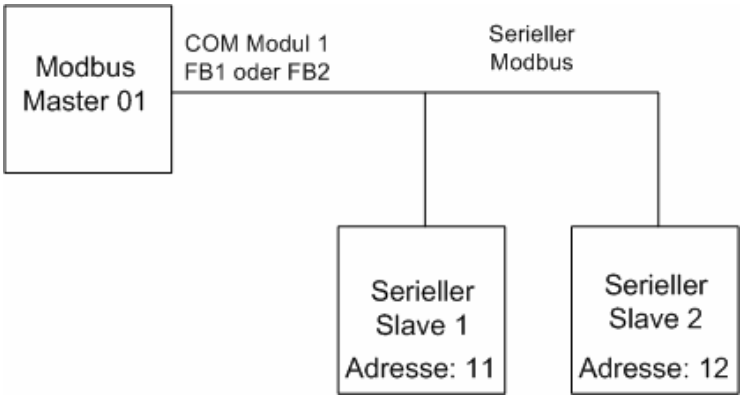


Bild 51: Serieller Modbus

Der HIMA Modbus Master unterstützt die Datenübertragung im RTU-Format (Remote Terminal Unit).

Der RTU Telegrammrahmen beginnt und endet mit den vom Anwender vorgegebenen Idle-Zeichen (Standardwert: 5 Idle-Zeichen).

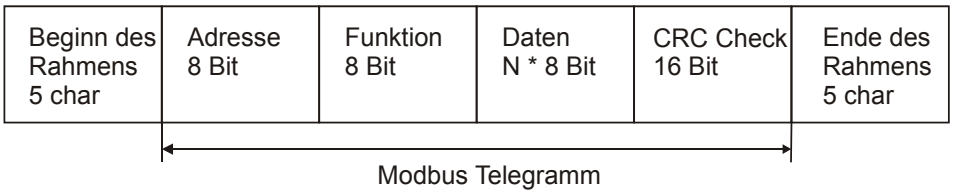


Bild 52: Modbus Telegramm

So erstellen Sie im Modbus Master einen seriellen Modbus:

1. Im Strukturbaum **Ressource**, **Protokolle**, **Modbus Master**, **Serieller Modbus** öffnen.
2. Rechtsklick auf **seriellen Modbus** und im Kontextmenü **Neu** wählen.
3. Aus der Liste **Modbus Slave** wählen und mit **OK** bestätigen.
4. Konfiguration des Modbus Slaves im Modbus Master:
   Edit zum Zuweisen der Systemvariablen wählen, siehe 7.5.14.
   Eigenschaften zum Konfigurieren der Eigenschaften wählen, siehe Kapitel 7.5.15.

7.5.13 Eigenschaften Serieller Modbus

Zur Konfiguration des seriellen Modbus im Modbus Master müssen die folgenden Parameter eingestellt werden.

Element	Beschreibung
Typ	Serieller Modbus
Name	Name des seriellen Modbus ist durch den Anwender wählbar
Beschreibung	Beliebige, eindeutige Beschreibung für den seriellen Modbus
Schnittstelle	Die Feldbusschnittstelle, die für den Modbus Master benutzt werden soll (fb1, fb2).
Baudrate [bps]	Übertragungsgeschwindigkeit für RS485 mögliche Werte: 38400 bit/s 19200 bit/s 9600 bit/s 4800 bit/s 2400 bit/s 1200 bit/s 600 bit/s 300 bit/s Standardwert: 38400
Parität	keine ungerade gerade Standardwert: gerade
Stop-Bits	Standard (passt die Anzahl der Stop-Bits der Parität an: mit Parität = 1 Stop-Bit, keine Parität 2 Stop-Bit.) ein Stop-Bit zwei Stop-Bit Standardwert: Standard
Anzahl Idle-Zeichen	Die Anzahl der Idle-Zeichen am Beginn und Ende des RTU-Telegrammrahmens. Wertebereich: 0 ... 65535 Standardwert: 5 Zeichen

Tabelle 180: Parameter serieller Modbus Master

7.5.14 Systemvariablen Modbus Slave

Im Editor Edit stehen die drei Statusvariablen (Systemvariablen) bereit.

Element	Beschreibung
Modbus-Slave Aktivierungssteuerung	Modbus Slave im Anwenderprogramm aktivieren oder deaktivieren. 0: Aktivieren 1: Deaktivieren
Modbus-Slave Fehler	Parameter wie bei TCP/UDP Slave Kapitel 7.5.7.
Modbus-Slave Zustand	Verbindungsstatus des Modbus Slave: 0: Deaktiviert 1: Nicht verbunden 2: Verbunden

Tabelle 181: Systemvariablen Modbus Slave

7.5.15 Eigenschaften Modbus Slave

Zur Konfiguration der Verbindung zu den seriellen Slaves müssen im Modbus Master die folgenden Parameter eingestellt werden.

Element	Beschreibung
Typ	Modbus Slave
Name	Name des Modbus Slave; durch den Anwender wählbar
Beschreibung	Beliebige, eindeutige Beschreibung für den Modbus Slave
Slave-Adresse	1 ... 247
Restliche Parameter wie bei TCP/UDP Slave, siehe Kapitel 7.5.7.	

Tabelle 182: Verbindungsparameter Modbus Master

i Die Receive Timeout ist beim seriellen Modbus Slave abhängig von der eingestellten Übertragungsgeschwindigkeit.

Ist die Baudrate 19200 [bit/s] oder höher, kann der Vorgabewert für die Receive Timeout verwendet werden. Bei niedrigeren Baudraten als 19200 [bit/s] muss die Receive Timeout erhöht werden.

7.6 Control-Panel (Modbus Master)

Im Control-Panel kann der Anwender die Einstellungen des Modbus Master überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. Master-Zustand usw.) des Masters angezeigt.

So öffnen Sie das Control Panel zur Überwachung des Modbus Master:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **Modbus Master** wählen.

7.6.1 Kontextmenü (Modbus Master)

Aus dem Kontextmenü des selektierte Modbus Master können die folgenden Kommandos gewählt werden:

Offline

Mit diesem Kommando wird der Modbus Master gestoppt.

Operate

Mit diesem Kommando wird der Modbus Master gestartet.

Statistik zurücksetzen

Setzt die statistischen Daten (z. B. Anzahl Busfehler, Zykluszeit min, max usw.) auf null zurück.

7.6.2 Anzeigefeld (Modbus Master)

In dem Anzeigefeld werden die folgenden Werte des selektierten Modbus Master angezeigt.

Element	Beschreibung
Name	Name des Modbus Masters
Master-Zustand	Der Modbus Master Zustand zeigt den momentanen Protokollzustand an: OPERATE OFFLINE
Anzahl Busfehler	Zähler Anzahl der Busfehler
Gestörte Verbindungen	Zähler Anzahl der gestörte Verbindungen
CPU-Last (projektierte)	Siehe Kapitel 7.4.2
CPU-Last (tatsächliche)	

Tabelle 183: Anzeigefeld Modbus Master

7.7 Control-Panel (Modbus Master->Slave)

Im Control-Panel kann der Anwender die Einstellungen der Kommunikationspartner des Modbus Master überprüfen und aktivieren/deaktivieren. Zudem werden aktuelle Statusinformationen (z. B. Slave-Zustand usw.) des Kommunikationspartners angezeigt.

So öffnen Sie das Control Panel zur Überwachung der Modbus Verbindung:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **Modbus Master, Slave** wählen.

7.8 Funktion der FBx LED beim Modbus Master

Das COM-Modul signalisiert den Zustand des lokalen Modbus Master-Protokolls mittels einer der jeweiligen Feldbusschnittstelle zugeordneten LEDs. Die Zustände dieser LEDs sind in der folgenden Tabelle dargestellt:

FBx LED	Farbe	Beschreibung
AUS	Gelb	Das Modbus Master Protokoll ist nicht aktiv! D.h. die Steuerung ist im Zustand STOPP oder es ist kein Modbus Master konfiguriert.
Blinkt	Gelb	Das Modbus Master Protokoll ist aktiv und befindet sich im Datenaustausch mit den Modbus Slaves.
AUS	Rot	Modbus Master Protokoll ist ohne Störung.
Blinkt	Rot	Folgende Ereignisse führen zur Störung. ▪ Fehlerhafte Antwort oder Fehlermeldung vom Slave empfangen ▪ Timeout zu einem oder mehreren Slaves ▪ Überschreitung des Rechenzeitbugets Tritt länger als 5 Sekunden kein Fehlerereignis auf, so wechselt die Anzeige in den Zustand „Protokoll ohne Störung“.

Tabelle 184: LED FBx beim Modbus Master

7.9 HIMA Modbus Slave

Der HIMA Modbus Slave kann gleichzeitig über die serielle Schnittstelle (RS485) und über Ethernet (TCP/UDP) mehrere Modbus Master bedienen.

Benötigte Ausstattung und Systemanforderungen

Element	Beschreibung
HIMA Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für Modbus TCP nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT D-SUB Anschlüsse FB1 und FB2 Wird Modbus RTU verwendet, muss das COM-Modul an den verwendeten seriellen Feldbusschnittstelle (FB1 oder FB2) mit einem optionalen HIMA RS485 Submodul ausgerüstet sein. Schnittstellenbelegung, siehe Kapitel 3.7.
Aktivierung	Jede der beiden Modbus Slave Funktionen muss einzeln freigeschaltet werden, siehe Kapitel 3.5. Modbus Slave RTU (RS485) Modbus Slave TCP

Tabelle 185: Ausstattung und Systemanforderung HIMA Modbus Slave

Modbus Slave (Eigenschaften)

Element	Beschreibung
Modbus Slave	Für jedes COM-Modul kann ein Modbus Slave konfiguriert werden.
Redundanz	Maximal 10 redundante Modbus Slave Kommunikationsmodul-Paare können in einem HIMax System betrieben werden. Solange ein Modbus Slave Kommunikationsmodul-Paar redundant arbeitet, werden die gleichen Eingangs- und Ausgangsdaten über beide Kommunikationsmodule mit dem Modbus-Master ausgetauscht.
Anzahl Masterzugriffe	RTU: Aufgrund der RS485 Übertagungstechnik kann nur ein Modbus Master auf den Slave zugreifen. TCP: Maximal 20 Modbus Master können auf den Slave zugreifen.
Max. Größe der Sendedaten	128 kB
Max. Größe der Empfangsdaten	128 kB
Darstellungsformat der Modbus-Daten	Die HIMax Steuerung verwendet das Big Endian Format für die Daten. Beispiel 32 Bit Daten (z. B. DWORD, DINT):

32 Bit Daten (hex)	0x12345678			
Speicher-Offset	0	1	2	3
Big Endian (HIMax)	12	34	56	78
Middle Endian (H51q)	56	78	12	34
Little Endian	78	56	34	12

Tabelle 186: Eigenschaften Modbus Slave

7.9.1 Konfiguration des Modbus TCP Slave

So legen Sie einen neuen HIMA Modbus Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **Modbus-Slave-Set** wählen, um einen neuen Modbus-Slave-Set hinzuzufügen.
3. Im Kontextmenü des Modbus-Slave-Set **Edit** wählen und **Eigenschaften Modbus-Slave-Set** öffnen, Standardwerte beibehalten.
4. Register **Modbus-Slave** wählen und folgende Einstellungen vornehmen:
   - **COM Modul** wählen
   - **TCP aktivieren** aktivieren
   - Die restlichen Parameter behalten die Standardwerte.

i Ein Konfigurationsbeispiel zur Verbindung eines HIMA Modbus TCP Slave mit einem HIMA Modbus TCP Master ist in Kapitel 7.2 beschrieben.

7.9.2 Konfiguration des redundanten Modbus TCP Slave

So legen Sie einen redundanten HIMA Modbus Slave an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle, Modbus-Slave-Set** öffnen.
2. Im Kontextmenü des Modbus-Slave-Set **Edit** wählen und **Eigenschaften Modbus-Slave-Set** öffnen und folgende Einstellung vornehmen:
   - **Redundanzbetrieb aktivieren** aktivieren.

   ☑ Das Register **Modbus-Slave-Redundant** wird automatisch hinzugefügt
3. Register **Modbus-Slave-Redundant** wählen und folgende Einstellungen vornehmen:
   - **COM Modul** wählen
   - **TCP aktivieren** aktivieren

Die restlichen Parameter behalten die Standardwerte.

i Die im Modbus-Slave-Set zugewiesenen Sende- und Empfangsvariablen gelten für beide Modbus-Slaves.

7.9.3 Regeln für den redundanten Modbus TCP Slave

Für den redundanten Betrieb von Modbus Slave Kommunikationsmodulen wird eine redundante HIMax Systemkonfiguration empfohlen, siehe Systemhandbuch HI 801 000 D. Ansonsten kann bereits beim Auftreten eines ersten Fehlers im HIMax System nicht sichergestellt werden, dass sich die Modbus Slave Kommunikationsmodul-Paare gegenüber ihrem externen Partner (Modbus Master) konsistent verhalten.

Zulässige Steckplätze der redundanten Modbus Slave COM-Module

Um mögliche Kollisionen auf dem HIMax Systembus zu minimieren, müssen die Systembus-Segmente (1 bis 3) auf dem Basisträger berücksichtigt werden. Daher sollten die redundanten Modbus Slave Kommunikationsmodule jeweils nur im gleichen Segmet eines Basisträgers auf den folgenden Steckplätzen gesteckt werden:

Segment	Steckplatz
1	3...6 (sofern hier keine Prozessormodule vorgesehen sind)
2	7...14
3	15...18

Tabelle 187: Zulässige Steckplätze der redundanten Modbus Slave COM-Module

Redundante Modbus Slave COM-Module in unterschiedlichen Basisträgers

Es dürfen nicht mehr als zwei redundante Modbus Slave Kommunikationsmodule betrieben werden, die sich in unterschiedlichen Basisträgern (0 bis 15) befinden.
Diese redundanten Modbus Slave Kommunikationsmodule dürfen sich dann auch nur in benachbarten Basisträgern befinden.

HINWEIS

Betriebsstörung möglich!

Steckplätze für redundante Modbus Slave Kommunikationsmodule nur entsprechend diesen Regeln verwenden!

7.10 Menüfunktionen des HIMA Modbus Slave Set

Die Menüfunktion Edit aus dem Kontextmenü des Modbus Slave Set öffnet den Dialog *Eigenschaften Modbus Slave Set*. Das Dialogfenster enthält die folgenden Register:

7.10.1 Eigenschaften Modbus Slave Set

Im Register *Eigenschaften Modbus Slave Set* werden die folgenden Parameter für den Modbus Slave eingestellt.

Element	Beschreibung
Name	Name des Modbus Slave Set
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld *Max. CPU Load [%]* übernehmen. Deaktiviert: Kein Limit der CPU-Last für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des COM-Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
Redundanzbetrieb aktivieren	Aktiviert: Redundanzbetrieb Deaktiviert: Monobetrieb Standardwert: Deaktiviert
Maximale Antwortzeit [ms]	Zeitspanne nach dem Empfang einer Anfrage, innerhalb welcher der Modbus Slave noch antworten darf. Wertebereich: 0-(2^{31}-1) ms Standardwert: 0 ms (0 = keine Limitierung)
Bereich zum Lesen der Funktionscodes 1 und 3	Der Parameter bestimmt, von welchem Datenbereich die Daten für den Funktionscode 1 und 3 gelesen werden sollen. Wertebereich: ▪ Importbereich ▪ Exportbereich (kompatibel zu 51q)
Bereich zum Lesen bei Funktionscode 23	Hier kann der Anwender den Bereich des Modbus Slave festlegen aus dem der Funktionscode 23 lesen soll. Importbereich: Der Master greift auf den Importbereich des Slaves lesend und schreibend zu. Exportbereich: Der Master liest vom Exportbereich des Slaves und schreibt auf den Importbereich des Slaves. Hinweis: Das Lesen und Schreiben erfolgt innerhalb von einem CPU-Zyklus. D.h. die gelesenen Daten wurden vom letzten CPU-Zyklus bereitgestellt.
Initialdaten bei Master Timeout	Nach Ablauf von Master Timeout wird der Verbindungsstatus auf nicht verbunden gesetzt. In Abhängigkeit von der folgenden Konfiguration werden die Input Variablen des Anwenderprogramms verarbeitet: Initialdaten annehmen: Eingangsdaten werden auf die Initialwerte zurückgesetzt. Letzten Wert beibehalten: Eingangsdaten behalten den letzten Wert.

Verhalten bei Verbindungsverlust in CPU verwenden	Mit diesem Parameter wird die Übernahme des Wertes aus *Initialdaten bei Master Timeout* gesteuert. **Wenn Sie ein Projekt von kleiner V3 konvertieren, muss dieser Wert deaktiviert werden.** Aktiviert: Der Wert aus *Initialdaten bei Master Timeout* wird übernommen. Deaktiviert: Eingangsdaten behalten den letzten Wert. Standardwert: Deaktiviert
Alternative Register/Bit Adressierung verwenden	Aktiviert Alternative Adressierung verwenden Deaktiviert Alternative Adressierung nicht verwenden Standardwert: Deaktiviert, siehe Kapitel 7.12.
Register Bereich Offset Bits Input	Wertebereich 0 ... 65535 Standardwert: 0
Register Bereich Offset Bits Output	Wertebereich 0 ... 65535 Standardwert: 0
Bit Bereich Offset Register Input	Wertebereich 0 ... 65535 Standardwert: 0
Bit Bereich Offset Register Output	Wertebereich 0 ... 65535 Standardwert: 0
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten des Protokolls zwischen COM und CPU ausgetauscht werden. Ist die *Refresh Rate* Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 0 ... (2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der gesamten Daten des Protokolls von der CPU zur COM, verteilt über mehrere CPU Zyklen zu je 1100 Byte pro Datenrichtung. Damit kann eventuell auch die Zykluszeit der Steuerung reduziert werden. Standardwert: Aktiviert

Tabelle 188: Register Eigenschaften Modbus Slave Set

7.10.2 Registervariable

(Register Zugriff)

Im Register Registervariablen sind die Variablen einzutragen, die der Master 16 bitweise adressiert (Funktionscode 3, 4, 6, 16, 23, 102, 103, 105, 106).

7.10.3 Bitvariablen

(Bit bzw. Coil-Zugriff)

In der Lasche Bitvariablen sind die Variablen einzutragen, die der Master 1 bitweise adressiert (Funktionscode 1, 2, 5, 15, 100, 101, 104).

7.10.4 Sende/Empfangsvariablen zuweisen

Im Register **Eingänge** werden alle Variablen, die der Modbus Slave vom Modbus Master empfängt, zugewiesen.

So konfigurieren Sie die Sendevariablen des Modbus Slave:

1. Im Strukturbaum den **Modbus Slave** wählen, den Sie konfigurieren wollen.
2. Rechtsklicken auf **Modbus Slave** und **Edit** wählen.
3. Register **Registervariablen** oder **Bitvariablen** wählen.
4. In der *Objektauswahl* eine **Variable** wählen und diese per Drag&Drop in den Bereich *Register Ausgänge* ziehen.
5. Diesen Schritt für jede weitere Variable die Sie als Sendevariable für den Modbus Slave definieren wollen wiederholen.
6. Rechtsklick in den Bereich *Register Ausgänge* und **Neue Offsets** wählen.

So konfigurieren Sie die Empfangsvariablen des Modbus Slave:

1. Im Strukturbaum den **Modbus Slave** wählen, den Sie konfigurieren wollen.
2. Klicken Sie mit der rechten Maustaste auf **Modbus Slave** und wählen Sie **Edit**.
3. Register **Registervariablen** oder **Bitvariablen** wählen.
4. In der *Objektauswahl* eine **Variable** wählen und diese per Drag&Drop in den Bereich *Register Eingänge* ziehen.
5. Diesen Schritt für jede weitere Variable die Sie als Empfangsvariable für den Modbus Slave definieren wollen wiederholen.
6. Rechtsklick in den Bereich *Register Eingänge* und **Neue Offsets** wählen.

7.10.5 Systemvariablen Modbus Slave Set

Das Register **Systemvariablen Modbus Slave Set** stellt die folgende Systemvariable bereit.

Element	Beschreibung
Redundanz-Zustand	Dieser Parameter beschreibt den Redundanz-Zustand des redundanten Modbus Slave Kommunikationsmodul-Paars. 0: redundante Modbus Slave COM-Module aktiv 1: Erstes Modbus Slave COM-Modul nicht aktiv 2: Redundantes Modbus Slave COM-Modul nicht aktiv 3: beide Modbus Slave COM-Module nicht aktiv

Tabelle 189: Anzeigefeld Modbus Master

7.10.6 Modbus Slave und Modbus Slave Redundant

Im Register **Modbus Slave** befinden sich die beiden Register Eigenschaften und Systemvariablen.

i Die Pin-Belegung der D-SUB-Anschlüsse (fb1, fb2) wird in den Datenblättern der jeweiligen HIMax Module beschrieben.

Eigenschaften

Element	Beschreibung
Modul	Auswahl des COM-Moduls, auf dem dieses Protokoll abgearbeitet wird.
Master Überwachungszeit [ms]	Timeoutzeit, innerhalb welcher der Slave mindestens einen Request von seinem Master erhalten haben muss. Empfängt der Slave innerhalb der Timeoutzeit keinen Request, so wird der “Master Verbindungsstatus” auf “nicht verbunden” gesetzt. Wertebereich 1...2^{31}-1 [ms] Standardwert: 0=Aus
Parameter der Ethernet Schnittstelle	
TCP aktivieren	Aktiviert TCP/IP-Verbindung aktiviert Deaktiviert TCP/IP-Verbindung deaktiviert Standardwert: Deaktiviert
TCP-Port	Standardwert: 502
Maximale Anzahl TCP-Verbindungen	Maximale Anzahl gleichzeitig offener TCP-Verbindungen als Server. Wertebereich: 1...20 Standardwert: 3
UDP aktivieren	Aktiviert UDP/IP-Verbindung aktiviert Deaktiviert UDP/IP-Verbindung deaktiviert Standardwert: Deaktiviert
UDP Port	Standardwert: 502
Parameter der seriellen Schnittstelle	
Name	Name der seriellen Schnittstelle
Schnittstelle	Auswahl der verfügbaren Feldbusschnittstellen, die für den Modbus Slave benutzt werden können (keine, fb1, fb2).
Slave-Adresse	Busadresse des Slave Wertebereich: 1 ... 247
Baudrate [bps]	Übertragungsgeschwindigkeit für RS485 mögliche Werte: 38400 bit/s 19200 bit/s 9600 bit/s 4800 bit/s 2400 bit/s 1200 bit/s 600 bit/s 300 bit/s
Parität	Wertebereich: ▪ keine ▪ ungerade ▪ gerade Standardwert: gerade
Stop-Bits	Wertebereich: Standard (passt die Anzahl der Stop-Bits der Parität an: mit Parität = 1 Stop-Bit, keine Parität 2 Stop-Bit.) ein Stop-Bit zwei Stop-Bit Standardwert: Standard
Anzahl Idle-Zeichen	Die Anzahl der Idle-Zeichen am Beginn und Ende des RTU-Telegrammrahmens. Wertebereich: 1 ... 65535 Standardwert: 5 Zeichen

Tabelle 190: Register TCP- und UDP-Ports für HIMA Modbus Slave

Das Register **Systemvariablen** stellt Systemvariablen bereit, die es erlauben, den Zustand des Modbus Slave im Anwenderprogramm auszuwerten und den Modbus Slave zu steuern

Element	Beschreibung
Durchschnittliche gleichzeitige Master Anfragen	Durchschnittliche Anzahl gleichzeitiger Master Anfragen
Gültige Master Anfragen	Anzahl der gültigen Master Anfragen seit dem letzten Rücksetzen aller Zähler oder Einschalten.
Master Anfragen	Gesamtanzahl aller Master Anfragen seit dem letzten Rücksetzen aller Zähler oder Einschalten.
Master Überwachungszeit [ms]	Timeoutzeit, innerhalb welcher der Slave mindestens einen Request von seinem Master erhalten haben muss. Empfängt der Slave innerhalb der Timeoutzeit keinen Request, so wird der *Master Verbindungsstatus* auf **1**, *nicht verbunden* gesetzt. In Abhängigkeit von der Konfiguration werden die Eingangsdaten des Anwenderprogramms auf Initialwerte zurückgesetzt oder die letzten Werte werden beibehalten.
Master Verbindungszustand	FALSE: nicht verbunden TRUE: Verbunden
Maximale gleichzeitige Master Anfragen	Maximale Anzahl gleichzeitiger Master Anfragen
Response timeouts	Anzahl der Response Timeouts seit dem letzten Rücksetzen aller Zähler oder Einschalten. Die Response Timeout ist die maximale Zeit, die bis zur Empfangsbestätigung einer Nachricht beim Absender vergehen darf.
Rücksetzen aller Zähler	Mit dieser Systemvariable können über das Anwenderprogramm alle Zähler zurückgesetzt werden. Ein Wechsel von 0 auf 1 löst die Reset-Funktion aus Werte > 1 werden als 1 behandelt.
Ungültige Master Anfragen	Anzahl ungültiger Master Anfragen seit dem letzten Rücksetzen aller Zähler oder Einschalten. Ungültige Anfragen sind solche, die der Modbus Slave mit einem Fehlercode an den Modbus Master beantwortet. Fehlerhafte Sendungen, die bereits auf Treiberebene erkannt und ausgefiltert werden (Framing Errors, CRC-Fehler, Längenfehler) sind hier nicht mit enthalten, sondern werden nur über die Diagnose gemeldet.
Verworfene Master Anfragen	Anzahl der verworfenen Master Anfragen seit dem letzten Rücksetzen aller Zähler oder Einschalten.

Tabelle 191: Register Systemvariablen für HIMA Modbus Slave

7.10.7 Modbus-Funktionscodes

Folgende Modbus-Funktionscodes werden vom HIMA Modbus Slave unterstützt.

Element	Code	Typ	Bedeutung
READ COILS	01	BOOL	Lesen mehrerer Variablen (BOOL) aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 251 Bytes
READ DISCRETE INPUT	02	BOOL	Lesen mehrerer Variablen (BOOL) aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 251 Bytes
READ HOLDING REGISTER	03	WORD	Lesen mehrerer Variablen beliebigen Typs aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 250 Bytes
READ INPUT REGISTER	04	WORD	Lesen mehrerer Variablen beliebigen Typs aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 250 Bytes
WRITE SINGLE COIL	05	BOOL	Schreiben eines einzelnen Signals (BOOL) in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 1 Byte
WRITE SINGLE REGISTER	06	WORD	Schreiben eines einzelnen Signals (WORD) in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 2 Bytes
Diagnostics	08	x	Nur Subcode 0: Loopback-Funktion des Slave
WRITE MULTIPLE COILS	15	BOOL	Schreiben mehrerer Variablen (BOOL) in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 247 Bytes
WRITE MULTIPLE REGISTER	16	WORD	Schreiben mehrerer Variablen beliebigen Typs in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 246 Bytes
READ WRITE MULTIPLE REGISTER	23	WORD	Schreiben und Lesen mehrerer Variablen beliebigen Typs in und aus dem Import-Bereich oder Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 242 Bytes (Anforderungstelegramm vom Modbus Master) 250 Bytes (Antwort an den Master).
Read Device Identification	43	x	Liefern die Identifikationsdaten des Slaves an den Master.

[1)]Export Bereich kann nur bei HIMA Slaves gewählt werden

Tabelle 192: Modbus-Funktionscodes des HIMA Modbus Slave

Die Funktionscodes 03, 04, 16 und 23 unterstützen neben dem Datentyp WORD (2 Byte) auch beliebige andere Datentypen.

Für jeden Request muss die Startadresse der ersten zu übertragenden Variable und die Anzahl der Register/Bits der zu übertragenden Variablen eingetragen werden.

Fehlercodes:

- Sendet der Master ein Telegramm mit unbekanntem Funktionscode, so antwortet die Steuerung mit Fehlercode 1 (Invalid Code).
- Stimmt das Telegramm des Masters nicht mit der Konfiguration der Modbus Slaves überein (d.h. z. B. das Anfragetelegramm endet nicht glatt an einer Variablengrenze), so antwortet der Slave mit Fehlercode 2 (Invalid Data).
- Sendet der Master ein Telegramm mit fehlerhaften Werten (z. B. Längenfeld), so antwortet der Slave mit Fehlercode 3 (Invalid Value).

Die Kommunikation erfolgt nur im Zustand RUN des COM Moduls. Master Anfragen in allen anderen Betriebszuständen des COM-Moduls werden nicht beantwortet.

Hinweis zur Modbus Funktion: Read Device Identification (43)

Der HIMax Modbus Slave liefert die Identifikationsdaten an den Master und unterstützt die folgenden Object-Ids:

Basic:
0x00 VendorName “HIMA Paul Hildebrandt GmbH + Co KG”
0x01 ProductCode “<Modul Seriennummer>”
0x02 MajorMinorRevision “<COM Vx.y CRC>”

Regular:
0x03 VendorUrl “http://www.hima.de”
0x04 ProductName “HIMax”
0x05 ModelName “HIMax”
0x06 UserApplicationName “--------[S.R.S]”

Extended:
0x80 leer “--------”
0x81 leer “--------”
0x82 leer “--------”
0x83 leer “--------”
0x84 leer “--------”
0x85 leer “--------“
0x86 CRC der Datei modbus.config “<0x234adcef>“
(Konfigurationsdatei des Modbus-Slave-Protokolls im Dateisystem der CPU. Zu vergleichen mit den Angaben in SILworX unter Online/Versionsvergleich).

Es werden folgende ReadDevice ID Codes unterstützt:
(1) Read Basic device identification (stream access)
(2) Read regular device identification (stream access)
(3) Read extended device identification (stream access)
(4) Read one specific identification object (inidividual access)

(Weitere Informationen zu Modbus finden Sie in der Spezifikation Modbus „Application Protocol Specification“ www.modbus.org)

7.10.8 HIMA spezifischen Funktionscodes

Die HIMA speziefischen Funktionscodes entsprechen den Standard Modbus Funktionscodes. Die einzigen Unterschiede sind die maximal zulässige Prozessdatenlänge von 1100 Bytes und das Format von Request und Responce-Header:

Element	Code	Typ	Bedeutung
Read Coils Extended	100 (0x64)	BOOL	Entspricht dem Functioncode 01 Lesen mehrerer Variablen (BOOL) aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Discrete Inputs Extended	101 (0x65)	BOOL	Entspricht dem Functioncode 02 Lesen mehrerer Variablen (BOOL) aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Holding Registers Extended	102 (0x66)	WORD	Entspricht dem Functioncode 03 Lesen mehrerer Variablen beliebigen Typs aus dem Import- oder Export[1)]-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read Input Registers Extended	103 (0x67)	WORD	Entspricht dem Functioncode 04 Lesen mehrerer Variablen beliebigen Typs aus dem Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Write Multiple Coils Extended	104 (0x68)	BOOL	Entspricht dem Functioncode 15 Schreiben mehrerer Variablen (BOOL) in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Write Multiple Registers Extended	105 (0x69)	WORD	Entspricht dem Functioncode 16 Schreiben mehrerer Variablen beliebigen Typs in den Import-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes
Read/Write Multiple Registers Extended	106 (0x6A)	WORD	Entspricht dem Functioncode 23 Schreiben und Lesen mehrerer Variablen beliebigen Typs in und aus dem Import-Bereich oder Export-Bereich des Slaves. Maximale Länge der Prozessdaten: 1100 Bytes (Anforderungstelegramm vom Modbus Master) 1100 Bytes (Antwort an den Master).

Format der Request und Response Header

Die Request und Response Header der HIMA speziefischen Modbus-Funktionscodes sind wie folgt aufgebaut:

Code	Request	Response
100 (0x64)	1 byte Functionscode 0x64 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260)	1 byte Functionscode 0x64 2 bytes Anzahl von Bytes = N N bytes Coil-Daten (8 Coils werden in ein Byte gepackt)
101 (0x65)	1 byte Functionscode 0x65 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x226)	1 byte Functionscode 0x65 2 bytes Anzahl von Bytes = N N bytes Coil-Daten (8 Coils werden in ein Byte gepackt)
102 (0x66)	1 byte Functionscode 0x66 2 bytes Startadresse 2 bytes Anzahl von Register 1 ... 550(0x226)	1 byte Functionscode 0x66 2 bytes Anzahl von Bytes = N N bytes Register-Daten
103 (0x67)	1 bytes Functionscode 0x67 2 bytes Startadresse 2 bytes Anzahl von Register 1 ... 550(0x226)	1 byte Functionscode 0x67 2 bytes Anzahl von Bytes = N N bytes Register-Daten
104 (0x68)	1 byte Functionscode 0x68 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260) 2 bytes Anzahl von Bytes = N N bytes Coil-Daten	1 byte Functionscode 0x66 2 bytes Startadresse 2 bytes Anzahl von Coils 1 ... 8800(0x2260)
105 (0x69)	1 byte Functionscode 0x69 2 bytes Startadresse 2 bytes Anzahl von Registern 1 ... 550(0x226) 2 bytes Anzahl von Bytes = N N bytes Register-Daten	1 byte Functionscode 0x69 2 bytes Startadresse 2 bytes Anzahl von Registern 1 ... 550(0x226)
106 (0x6A)	1 byte Functionscode 0x6a 2 bytes Lese-Startadresse 2 bytes Anzahl von Leseregistern 1 ... 550(0x226) 2 bytes Schreib-Startadresse 2 bytes Anzahl von Schreibregistern 1 ... 550(0x226) 2 bytes Anzahl von Bytes zum Schreiben=N N bytes Register-Daten	1 byte Functionscode 0x6a 2 bytes Anzahl von Bytes = N N bytes Register-Daten

7.11 Modbus Adressierung durch Bit und Register

Dieser Adressierungsmodus entspricht dem Standard der Modbus Adressierung und kennt nur die zwei Datenlängen Bit (1 Bit) und Register (16 Bit) mit denen alle zugelassenen Datentypen übertragen werden können.

Im Modbus Slave gibt es einen **Register Bereich** (Ein- und Ausgänge) und einen **Bit Bereich** (Ein- und Ausgänge). Beide Bereiche sind voneinander getrennt und können alle zugelassenen Datentypen aufnehmen. Der Unterschied dieser Bereiche besteht in den erlaubten Modbus-Funktionscodes, mit denen auf diese Bereiche zugegriffen werden kann.

i Die Modbus Adressierung durch Bit und Register garantiert keine Variablen-Integrität, d. h. mit diesem Zugriff können beliebige Teile von Variablen gelesen/geschrieben werden. Variablen vom Typ BOOL werden gepackt abgelegt, d. h. jede Variable vom Typ BOOL ist als Bit innerhalb eines Byte abgelegt.

7.11.1 Register Bereich

In der Lasche **Registervariablen** werden die Variablen im Register Bereich angelegt. Weitere Informationen zur Zuweisung der Sende/Empfangsvariablen siehe Kapitel 7.10.4.

i Um mit den Modbus-Funktionscodes 1, 2, 5, 15 auf die Variablen im Register Bereich zuzugreifen, müssen die Variablen in den Bit Bereich gespiegelt werden, siehe Kapitel 7.12.1.

Auf die Variablen im Register Bereich kann nur über die Modbus-Funktionscodes 3, 4, 6, 16, 23 zugegriffen werden. Dazu muss in den Eigenschaften des Funktionscodes die Startadresse der ersten Variable eingetragen werden.

Beispiel: Zugriff auf Variablen im Register Bereich des Modbus Slave

Register Variablen	Register.Bit	Bit
00_Register_Bereich_WORD	0.0	0
01_Register_Bereich_SINT	1.8	16
02_Register_Bereich_SINT	1.0	24
03_Register_Bereich_REAL	2.0	32
04_Register_Bereich_BOOL	4.8	64
05_Register_Bereich_BOOL	4.9	65
06_Register_Bereich_BOOL	4.10	66
07_Register_Bereich_BOOL	4.11	67
08_Register_Bereich_BOOL	4.12	68
09_Register_Bereich_BOOL	4.13	69
10_Register_Bereich_BOOL	4.14	70
11_Register_Bereich_BOOL	4.15	71
12_Register_Bereich_BOOL	4.0	72
13_Register_Bereich_BOOL	4.1	73
14_Register_Bereich_BOOL	4.2	74
15_Register_Bereich_BOOL	4.3	75
16_Register_Bereich_BOOL	4.4	76
17_Register_Bereich_BOOL	4.5	77
18_Register_Bereich_BOOL	4.6	78
19_Register_Bereich_BOOL	4.7	79

Tabelle 193: Register Variablen im Register Bereich des Modbus Slave

HIMA Modbus Master Konfiguration des Anforderungstelegramms

So lesen Sie im Modbus Master die Variablen 01_Register_Bereich_SINT bis 03_Register_Bereich_REAL ein:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Read Holding Registers (3)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (3)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Lesebereichs, 1** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (3)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Eingangsvariablen** ziehen.

Register Variablen	Offset
01_Register_Bereich_SINT	0
02_Register_Bereich_SINT	1
03_Register_Bereich_REAL	2

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

7.11.2 Bit Bereich

In der Lasche **Bitvariablen** werden die Variablen im Bit Bereich angelegt. Weitere Informationen zur Zuweisung der Sende/Empfangsvariablen siehe Kapitel 7.10.4.

i Um mit Modbus-Funktionscodes 3, 4, 6, 16, 23 auf die Variablen im Bit Bereich zuzugreifen, müssen die Variablen in den Register Bereich gespiegelt werden, siehe Kapitel 7.12.2.

Auf die Variablen im Bit Bereich kann nur über die Modbus-Funktionscodes 1, 2, 5, 15 zugegriffen werden. Dazu muss in den Eigenschaften des Funktionscodes die Startadresse der ersten Variable eingetragen werden.

Beispiel: Zugriff auf Variablen im Bit Bereich des Modbus Slave

Bit Variablen	Bit	Register.Bit
00_BIT_Bereich_WORD	0	0.0
01_BIT_Bereich_SINT	16	1.8
02_BIT_Bereich_SINT	24	1.0
03_BIT_Bereich_REAL	32	2.0
04_BIT_Bereich_BOOL	64	4.8
05_BIT_Bereich_BOOL	65	4.9
06_BIT_Bereich_BOOL	66	4.10
07_BIT_Bereich_BOOL	67	4.11
08_BIT_Bereich_BOOL	68	4.12
09_BIT_Bereich_BOOL	69	4.13
10_BIT_Bereich_BOOL	70	4.14
11_BIT_Bereich_BOOL	71	4.15
12_BIT_Bereich_BOOL	72	4.0
13_BIT_Bereich_BOOL	73	4.1
14_BIT_Bereich_BOOL	74	4.2
15_BIT_Bereich_BOOL	75	4.3
16_BIT_Bereich_BOOL	76	4.4
17_BIT_Bereich_BOOL	77	4.5
18_BIT_Bereich_BOOL	78	4.6
19_BIT_Bereich_BOOL	79	4.7

Tabelle 194: Bit Variablen im Bit Bereich des Modbus Slave

HIMA Modbus Master Konfiguration des Anforderungstelegramms

So Lesen Sie im Modbus Master die Variablen 04_BIT_Bereich_BOOL bis 06_Bereich _BOOL ein:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Read Coils (1)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Read Coils (1)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Lesebereichs, 64** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Read Coils (1)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Eingangsvariablen** ziehen.

Bit Variablen	Offset
04_BIT_Bereich_BOOL	0
05_BIT_Bereich_BOOL	1
06_BIT_Bereich_BOOL	2

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

7.12 Offsets für alternative Modbus Adressierung

Um mit den Modbus-Funktionscodes (Typ Register) auf Variablen im **Bit Bereich** und mit den Modbus-Funktionscodes (Typ Bit) auf Variablen im **Register Bereich** zuzugreifen, müssen die Variablen in den jeweils anderen Bereich gespiegelt werden.
In der Lasche **Eigenschaften/Offsets** werden die Offsets der gespiegelten Variablen eingetragen.

So spiegeln Sie die Variablen in den Bit- und Register Bereich:

1. Im Kontextmenü des Modbus Slave **Edit**, **Offsets** wählen und **Alternative Register/Bit-Adressierung verwenden** aktivieren.
   ☑ Damit werden die Variablen in den jeweils anderen Bereich gespiegelt.
2. Offset für die gespiegelten Variablen im Bit- und Register Bereich eintragen.

i Die gespiegelten Variablen im Bit-/ Register Bereich und die vorhandenen Variablen im Bit-/ Register Bereich dürfen sich bzgl. der Modbus Adressen nicht überlappen.

Element	Beschreibung/ Wertebereich
Alternative Register/Bit-Adressierung verwenden	Aktiviert: Alternative Adressierung verwenden Deaktiviert: Alternative Adressierung nicht verwenden Standardwert: Deaktiviert
Register Bereich Offset/Bit Eingänge	0 ... 65535
Register Bereich Offset/Bit Ausgänge	0 ... 65535
Bit Bereich Offset/Register Eingänge	0 ... 65535
Bit Bereich Offset/Register Ausgänge	0 ... 65535

Tabelle 195: Lasche Offsets für HIMA Modbus Slave

7.12.1 Zugriff auf die Registervariablen im Bit Bereich des Modbus Slave

Um mit den Modbus-Funktionscodes (Typ Bit) 1, 2, 5, 15 auf die Register Variablen zuzugreifen, müssen die Register Variablen in den **Bit Bereich** gespiegelt werden.
In der Lasche **Eigenschaften/Offsets** müssen die Offsets der gespiegelten Register Variablen eingetragen werden.

Beispiel:

Bit Bereich Offset/Register Eingänge 8000
Bit Bereich Offset/Register Ausgänge 8000
Hier liegen die aus dem Register Bereich in den Bit Bereich gespiegelten Variablen ab Bit Adresse 8000.

Gespiegelte Register Variablen	Bit
00_Register_Bereich_WORD	8000
01_Register_Bereich_SINT	8016
02_Register_Bereich_SINT	8024
03_Register_Bereich_REAL	8032
04_Register_Bereich_BOOL	8064
05_Register_Bereich_BOOL	8065
06_Register_Bereich_BOOL	8066
07_Register_Bereich_BOOL	8067
08_Register_Bereich_BOOL	8068
09_Register_Bereich_BOOL	8069
10_Register_Bereich_BOOL	8070
11_Register_Bereich_BOOL	8071
12_Register_Bereich_BOOL	8072
13_Register_Bereich_BOOL	8073
14_Register_Bereich_BOOL	8074
15_Register_Bereich_BOOL	8075
16_Register_Bereich_BOOL	8076
17_Register_Bereich_BOOL	8077
18_Register_Bereich_BOOL	8078
19_Register_Bereich_BOOL	8079

Tabelle 196: Aus dem Register Bereich in den Bit Bereich gespiegelte Variablen

HIMA Modbus Master Konfiguration des Anforderungstelegramms

So lesen Sie im Modbus Master die Variablen 04_Register_Bereich_BOOL bis 06_Register_Bereich_BOOL ein:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Read Coils (1)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Read Coils (1)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Lesebereichs, 8064** eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Read Coils (1)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Eingangsvariablen** ziehen.

Gespiegelte Register Variablen	Offset
04_Register_Bereich_BOOL	0
05_Register_Bereich_BOOL	1
06_Register_Bereich_BOOL	2

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

7.12.2 Zugriff auf die Bitvariablen im Registerbereich des Modbus Slave

Um mit den Modbus-Funktionscodes (Typ Register) 3, 4, 6, 16, 23 auf die Bit Variablen zuzugreifen, müssen die Bit Variablen in den **Register Bereich** gespiegelt werden.
In der Lasche **Eigenschaften/Offsets** müssen die Offsets der gespiegelten Bit Variablen eingetragen werden.

Beispiel:

Register Bereich Offset/Bit Eingänge: 1000
Register Bereich Offset/Bit Ausgänge: 1000
Hier liegen die aus dem Bit Bereich in den Register Bereich gespiegelten Variablen ab Register Adresse 1000.

Gespiegelte Bit Variablen	Register.Bit
00_BIT_Bereich_WORD	1000.0
01_BIT_Bereich_SINT	1001.8
02_BIT_Bereich_SINT	1001.0
03_BIT_Bereich_REAL	1002.0
04_BIT_Bereich_BOOL	1004.8
05_BIT_Bereich_BOOL	1004.9
06_BIT_Bereich_BOOL	1004.10
07_BIT_Bereich_BOOL	1004.11
08_BIT_Bereich_BOOL	1004.12
09_BIT_Bereich_BOOL	1004.13
10_BIT_Bereich_BOOL	1004.14
11_BIT_Bereich_BOOL	1004.15
12_BIT_Bereich_BOOL	1004.0
13_BIT_Bereich_BOOL	1004.1
14_BIT_Bereich_BOOL	1004.2
15_BIT_Bereich_BOOL	1004.3
16_BIT_Bereich_BOOL	1004.4
17_BIT_Bereich_BOOL	1004.5
18_BIT_Bereich_BOOL	1004.6
19_BIT_Bereich_BOOL	1004.7

Tabelle 197: Aus dem Bit Bereich in den Register Bereich gespiegelte Variablen

HIMA Modbus Master Konfiguration des Anforderungstelegramms

So Lesen Sie im Modbus Master die Variablen 01_BIT_Bereich_SINT bis 03_BIT_Bereich_REAL ein:

1. Rechtsklick auf **TCP/UDP-Slave** und im Kontextmenü **Neu** wählen.
2. Aus der Liste das Anforderungstelegramm **Read Holding Registers (3)** wählen.
3. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (3)** und im Kontextmenü **Eigenschaften** wählen.
   - **Startadresse des Lesebereichs** 1001 eintragen.
4. Rechtsklick auf das Anforderungstelegramm **Read Holding Registers (3)** und im Kontextmenü **Edit** wählen.
5. In der **Objektauswahl** die folgenden Variablen auswählen und diese per Drag&Drop in das Register **Eingangsvariablen** ziehen.

Gespiegelte Bit Variablen	Offset
01_BIT_Bereich_SINT	0
02_BIT_Bereich_SINT	1
03_BIT_Bereich_REAL	2

6. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

7.13 Control-Panel (Modbus Slave)

Im Control-Panel kann der Anwender die Einstellungen des Modbus Slave überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. Master-Zustand usw.) des Slaves angezeigt.

So öffnen Sie das Control Panel zur Überwachung des Modbus Slave:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **Modbus Slave** wählen.

7.13.1 Kontextmenü (Modbus Slave)

Aus dem Kontextmenü des selektierte Modbus Slave kann das folgende Kommando gewählt werden:

Zurücksetzen Statistik

Setzt die statistischen Daten (Zykluszeit min, max usw.) auf null zurück.

7.13.2 Anzeigefeld (Modbus Slave)

Im Anzeigefeld werden die folgenden Werte des selektierte Modbus Slave angezeigt.

Element	Beschreibung
Name	Name des Modbus Slave
CPU-Last (projektierte) [%]	Siehe Kapitel 7.10
CPU-Last (tatsächliche) [%]	

Tabelle 198: Anzeigefeld Modbus Slave

7.13.3 Anzeigefeld (Masterdaten)

In dem Anzeigefeld Masterdaten werden die folgenden Werte angezeigt.

Element	Beschreibung
Name	Name der Masterdaten
Schnittstelle	Modbusfähige Schnittstelle des COM-Moduls (RS485, Ethernet UDP/TCP)
Anfragen	Gesamtanzahl aller Master Anfragen seit dem letzten Zähler Reset.
Gültige Anfragen	Anzahl gültiger Master Anfragen seit dem letzten Zähler Reset.
Ungültige Anfragen	Anzahl ungültiger Master Anfragen seit dem letzten Zähler Reset.
Timeout [ms]	Timeoutzeit, innerhalb welcher der Slave mindestens einen Request von seinem Master erhalten haben muss. Empfängt der Slave innerhalb der Timeoutzeit keinen Request, so wird der *Master Verbindungsstatus* auf *nicht verbunden* gesetzt.
Verbindungszustand	0=nicht überwacht (Master Request Timeout ist Null) 1=nicht verbunden 2=verbunden

Tabelle 199: Anzeigefeld Masterdaten

7.14 Funktion der FBx LED beim Modbus Slave

Das COM-Modul signalisiert den Zustand des lokalen Modbus-Slave-Protokolls mittels einer der den jeweiligen Feldbusschnittstelle zugeordneten LEDs. Die Zustände dieser LEDs sind in der folgenden Tabelle dargestellt:

FBx LED	Farbe	Beschreibung
AUS	Gelb	Das Modbus-Slave-Protokoll ist nicht aktiv! Das heißt, die Steuerung ist im Zustand STOPP oder es ist kein Modbus Master konfiguriert.
Blinkt	Gelb	Das Modbus-Slave-Protokoll ist aktiv und befindet sich im Datenaustausch mit dem Modbus Master.
AUS	Rot	Modbus-Slave-Protokoll ist ohne Störung.
Blinkt	Rot	Modbus-Slave-Protokoll ist gestört. Folgende Ereignisse führen zur Störung. ▪ Unbekannter Funktionscode empfangen ▪ Anfrage mit falscher Adressierung empfangen ▪ Überschreitung des Rechenzeitbugets Tritt länger als 5 Sekunden kein Fehlerereignis auf, so wechselt die Anzeige in den Zustand *Protokoll ohne Störung*.

Tabelle 200: LED FBx beim Modbus Slave

7.14.1 Fehlercodes der Modbus TCP/IP-Verbindung

Die Fehlercodes der Modbus TCP/IP-Verbindung werden im Dialogfenster **Diagnose** angezeigt.

Fehlercode	Beschreibung
35	Operation ist blockiert
48	Port ist bereits in Verwendung
50	Netzwerk läuft nicht
53	Verbindung durch Software abgebrochen
54	Verbindung durch Partner abgebrochen
55	Kein Pufferspeicher mehr verfügbar
60	Timeout aufgetreten Verbindung geschlossen
61	Verbindung abgewiesen (durch Partner)
65	Kein Routing-Eintrag zum Partner vorhanden

Tabelle 201: Fehlercodes Modbus TCP/IP

8 Send & Receive TCP

S&R TCP ist ein herstellerunabhängiges Standardprotokoll für zyklischen und azyklischen Datenaustausch und verwendet außer TCP/IP kein spezielles Protokoll.

Mit dem S&R-TCP-Protokoll unterstützt die HIMax Steuerung nahezu jedes Fremdsystem sowie PC's mit vorhandener Socket–Schnittstelle (z. B. Winsock.dll) zu TCP/IP.

S&R TCP ist kompatibel zu der Siemens SEND/RECEIVE Schnittstelle und erlaubt die Kommunikation mit Siemens Steuerungen über TCP/IP. Der Datenaustausch erfolgt über die S7-Funktionsbausteine AG_SEND (FC5) und AG_RECV (FC6).

8.1 Systemanforderung

Benötigte Ausstattung und Systemanforderung:

Element	Beschreibung
Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für Send & Receive TCP nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT Es kann für jedes COM-Modul ein S&R-TCP-Protokoll konfiguriert werden.
Aktivierung	Die Freischaltung erfolgt per Software-Freischaltcode, siehe Kapitel 3.5.

Tabelle 202: Systemanforderung und Ausstattung S&R TCP

Eigenschaften des S&R-TCP-Protokolls:

Element	Beschreibung
Sicherheitsgerichtet	Nein
Datenaustausch	Zyklischer und azyklischer Datenaustausch über TCP/IP.
Funktionsbausteine	Die S&R-TCP-Funktionsbausteine müssen beim azyklischen Datenaustausch verwendet werden.
TCP-Verbindungen	Bis zu 32 TCP-Verbindungen können in einer Steuerung konfiguriert werden, sofern nicht die maximale Größe der Sendedaten oder Empfangsdaten überschritten wird.
Max. Größe der Prozessdaten	Insgesamt können 128 kB Daten gesendet und 128 kB Daten empfangen werden. i Um die maximale Anzahl der Nutzdaten zu ermitteln, müssen alle Statusvariablen der verwendeten TCP-Verbindungen und der TCP/SR Funktionsbausteine von der maximalen Anzahl Sendedaten (128 kB) abgezogen werden. Die Aufteilung auf die einzelnen TCP-Verbindungen ist beliebig.

Tabelle 203: Eigenschaften S&R TCP

8.1.1 Anlegen eines S&R-TCP-Protokolls

So legen Sie ein neues S&R-TCP-Protokoll an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **Send/Receive over TCP** wählen, um ein neues S&R-TCP-Protokoll hinzuzufügen.
3. Im Kontextmenü vom Send/Receive-over-TCP-Protokoll **Eigenschaften**, **Allgemein** das **COM-Modul** auswählen.

8.2 Beispiel: S&R TCP Konfiguration

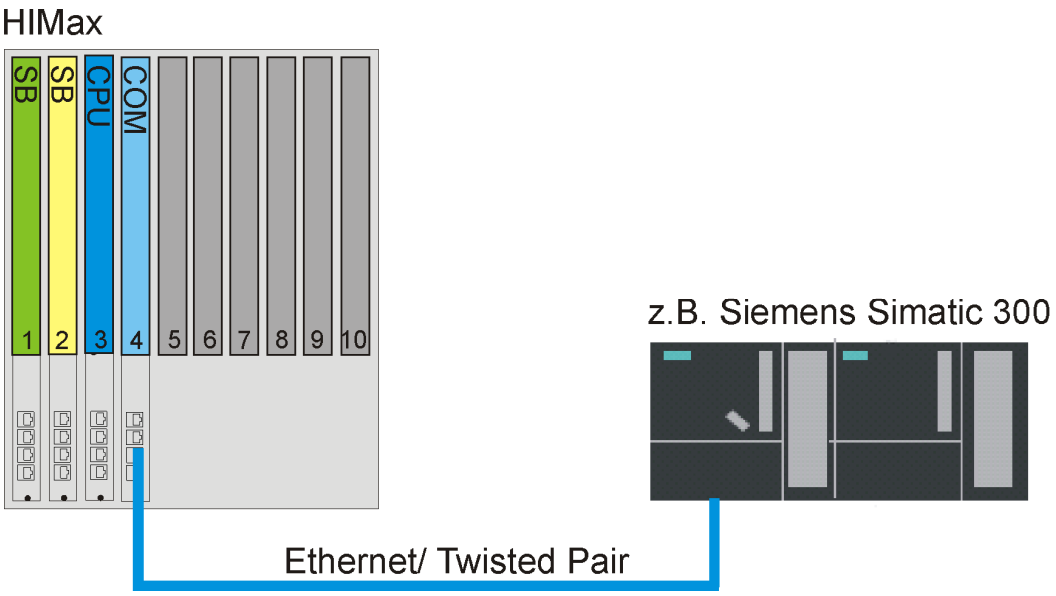


Bild 53: Verschaltung HIMax und Siemens Steuerung

In diesem Beispiel wird das Protokoll Send/Receive over TCP in einer HIMax Steuerung eingerichtet. Die HIMax soll zyklisch über S&R TCP mit einer Siemens Steuerung (z. B. SIMATIC 300) kommunizieren.

Dabei ist die HIMax (Client) die aktive Station, welche die TCP-Verbindung zur passiven Siemens SIMATIC 300 (Server) aufbaut. Nach dem Verbindungsaufbau sind aber beide Steuerungen gleichberechtigt und können jederzeit senden und empfangen.

Bei der Zusammenschaltung der HIMax und der Siemens SIMATIC 300 ist folgendes zu beachten:

Für die HIMax gelten die im Abschnitt 8.1 (Systemanforderungen) beschriebenen Anforderungen.

Die HIMax und die Siemens SIMATIC 300 werden über ihre Ethernet Schnittstellen miteinander verbunden.

Die HIMax und die Siemens SIMATIC 300 müssen sich im gleichen Subnet befinden oder bei Verwendung eines Routers die entsprechenden Routing Einträge besitzen.

In diesem Beispiel sollen zwei BYTES und ein WORD von der HIMax Steuerung zur Siemens SIMATIC 300 gesendet werden. Die Variablen werden in der SIMATIC 300 vom Baustein AG_RECV (FC 6) empfangen und intern an den Baustein AG_SEND (FC 5) übergeben. Über den Baustein AG_SEND (FC 5) sendet die SIMATIC 300 die Variablen unverändert an die HIMax Steuerung zurück.

Die Übertragung der Variablen kann der Anwender nach der Konfiguration mit dem HIMA Force-Editor prüfen.

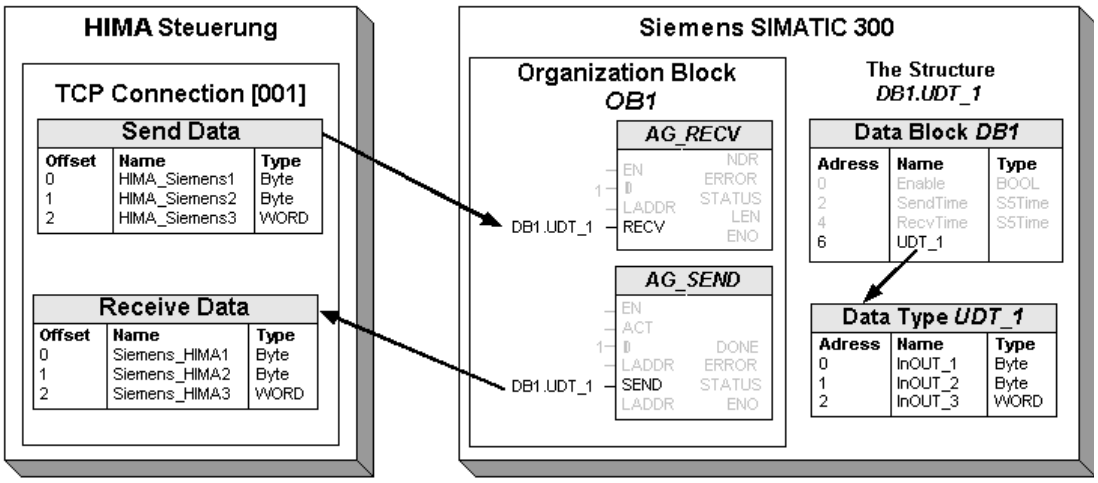


Bild 54: Datenübertragung zwischen HIMax und Siemens Steuerung

Konfigurationsbeschreibung HIMax Steuerung:

Element	Beschreibung
TCP-Verbindung [001]	In diesem Dialog stehen alle Parameter, die für die Kommunikation mit dem Kommunikationspartner (Siemens SIMATIC 300) notwendig sind.
Sendedaten	Die Offsets und Typen der Variablen in der Steuerung müssen mit der Adresse und den Typen der Variablen im Datentyp *UDT_1* der SIMATIC 300 übereinstimmen.
Empfangsdaten	Die Offsets und Typen der Variablen in der HIMax Steuerung müssen mit der Adresse und den Typen der Variablen im Datentyp *UDT_1* der SIMATIC 300 übereinstimmen.

Tabelle 204: Konfiguration HIMax Steuerung

Konfigurationsbeschreibung Siemens SIMATIC 300:

Element	Beschreibung
Organisationsbaustein *OB1*	Die Funktionsbausteine *AG_RECV* (FC6) und *AG_SEND* (FC 5) müssen im Organisationsbaustein *OB1* angelegt und konfiguriert werden.
AG_RECV (FC 6)	Der Funktionsbaustein *AG_RECV* (FC 6) übernimmt die empfangenen Daten vom Kommunikationspartner in den Datentyp *DB1.UDT_1*. Die Eingänge *ID* und *LADDR* müssen für die Kommunikation mit dem Kommunikationspartner entsprechend konfiguriert werden.
AG_SEND (FC 5)	Der Funktionsbaustein *AG_SEND* (FC 5) überträgt die Daten aus dem Datentyp *DB1.UDT_1* zum Kommunikationspartner. Die Eingänge *ID* und *LADDR* müssen für die Kommunikation mit dem Kommunikationspartner entsprechend konfiguriert werden.
Datenbaustein *DB1*	Der Datentyp *UDT_1* wird im Datenbaustein *DB1* definiert.
Datentyp *UDT_1*	Die Adressen und Typen der Variablen in der SIMATIC 300 müssen mit den Offsets und den Typen der Steuerung übereinstimmen. Der Datentyp *UDT_1* übernimmt die empfangenen Nutzdaten und speichert diese bis zur Übertragung an den Kommunikationspartner.

Tabelle 205: Konfiguration Siemens SIMATIC 300

8.2.1 S&R TCP Konfiguration der Siemens Steuerung SIMATIC 300

Die folgende Schrittanleitung zur Konfiguration der Siemens-Steuerung erhebt keinen Anspruch auf Vollständigkeit.

Alle Angaben sind ohne Gewähr, maßgebend zur Projektierung der Siemens-Steuerung ist die Dokumentation von Siemens.

So erstellen Sie den S&R-TCP-Server im Projekt der SIMATIC 300:

1. SIMATIC Manager starten.
2. Im SIMATIC Manager das Projekt der SIMATIC 300 Steuerung öffnen.
3. In diesem Projekt das *Industrial Ethernet* und die *MPI* Verbindungen Erstellen und Konfigurieren.

So erstellen Sie den Datentyp UDT1 mit den folgenden Variablen:

1. Ordner *Bausteine* im Siemens SIMATIC Manager öffnen.
2. Im Hauptmenü **Einfügen**, **S7 Baustein**, **Datentyp** öffnen und einen Datentyp erstellen.
3. Datentyp den Namen **UDT1** zuweisen
4. Datentyp den Symbolischen Namen **UDT_1** zuweisen.
5. Im Datentyp *UDT_1* die drei Variablen **InOut_x** wie im Bild unten erstellen.

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	InOut_1	BYTE	B#16#0	
+1.0	InOut_2	BYTE	B#16#0	
+2.0	InOut_3	WORD	W#16#0	
=4.0		END_STRUCT		

Bild 55: Variablenliste im Siemens-Baustein UDT1

Beim zyklischem und azyklischen Datenaustausch ist zu beachten, dass manche Steuerungen (z. B. SIMATIC 300) so genannte *Pad Bytes* einfügen. Damit wird sichergestellt, dass alle Datentypen, die größer als ein Byte sind, immer an einem geraden Offset beginnen und dass die Gesamtlänge aller definierten Variablen ebenfalls immer gerade ist.

In diesen Fällen müssen an den entsprechenden Stellen Dummy-Bytes in der HIMax Steuerung eingefügt werden (siehe Kapitel 8.6, Fremdsysteme mit Pad Bytes).

So erstellen Sie den Datenbaustein DB1 für die Funktionsbausteine FC 5 and FC 6:

1. Im Hauptmenü **Einfügen**, **S7 Baustein**, **Datenbaustein** wählen und Datenbaustein erstellen.
2. Datenbaustein den Namen **DB1** zuweisen.
3. Datenbaustein den Symbolischen Namen **DB1** zuweisen.
4. Dem Datenbaustein *DB1* den Datentyp *UDT_1* zuweisen.
5. Im Datenbaustein *DB1* die Datentypen wie im Bild unten dargestellt parametrieren.

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	Enable	BOOL	FALSE
+2.0	SendTime	S5TIME	S5T#100MS
+4.0	RecvTime	S5TIME	S5T#10MS
+6.0	UDT_1	"UDT_1"	
=10.0		END_STRUCT	

Bild 56: Variablenliste im Siemens-Baustein DB1

So erstellen Sie im Symboleditor die folgenden Symbole

1. Dialogfenster *KOP/AWL/FUP* mit einem Doppelklick auf den Organisationsbaustein **OB1** öffnen.
2. Im Hauptmenü **Extras**, **Symboltabelle** den Symboleditor öffnen.
3. Den *Symboleditor* mit den Variablen **M 1.0** ... **MW 5** wie im Bild unten dargestellt ergänzen.

S7-Programm(1) (Symbole) -- S7_Pro5\SIMATIC 300-Stati

	Status	Symbol	Adresse	Datentyp
1		Cycle Execution	OB 1	OB 1
2		DB1	DB 1	DB 1
3		UDT_1	UDT 1	UDT 1
4		RecDone	M 1.0	BOOL
5		RecError	M 1.1	BOOL
6		SendDone	M 1.2	BOOL
7		SendError	M 1.3	BOOL
8		RecStatus	MW 1	WORD
9		RecLen	MW 3	INT
10		SendStatus	MW 5	WORD
11				

Bild 57: SIMATIC-Symboleditor

So erstellen Sie den FC-Baustein *AG_RECV* (FC 6):

1. Dialogfenster *KOP/AWL/FUP* öffnen.
2. Folgende FC-Bausteine nacheinander aus der Struktur im linken Teil des Symatic-Managers auswählen:
   1 *Oder-Gatter*
   1 *S_VIMP*
   1 *AG_RECV (FC 6)*.
3. Diese *Funktionsbausteine* per Drag&Drop in den Organisationsbaustein *OB1* ziehen.
4. *Funktionsbausteine* wie im Bild unten dargestellt Verbinden und Konfigurieren.
5. Rechtsklick auf den FC-Baustein **AG_RECV (FC 6)** und **Eigenschaften** wählen.
6. **Aktiver Verbindungsaufbau** deaktivieren und Ports Konfigurieren.
7. Bausteinparameter *LADDR* notieren und diese im Funktionsplan am Baustein *AG_RECV (FC 6)* eintragen.

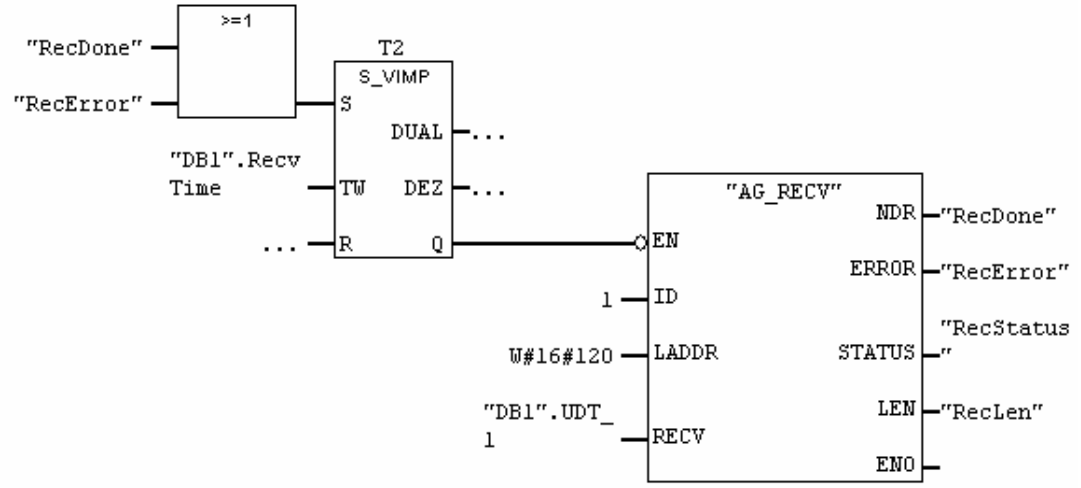


Bild 58: Funktionsplan zum Empfangen

So erstellen Sie den FC-Baustein AG_SEND (FC 5):

1. Dialogfenster *KOP/AWL/FUP* öffnen.
2. Folgende FC-Bausteine nacheinander aus der Struktur im linken Teil des Symatic-Managers auswählen:
   1 *Oder-Gatter*
   1 *S_VIMP*
   1 *AG_SEND (FC 5).*
3. Diese *Funktionsbausteine* per Drag&Drop in den Organisationsbaustein *OB1* ziehen.
4. *Funktionsbausteine* wie im Bild unten dargestellt verbinden und konfigurieren.
5. Rechtsklick auf den FC-Baustein **AG_SEND (FC 5)** und **Eigenschaften** wählen.
6. **Aktiver Verbindungsaufbau** deaktivieren und Ports Konfigurieren.
7. Bausteinparameter *LADDR* notieren und diese im Funktionsplan am Baustein *AG_SEND (FC 5)* eintragen.

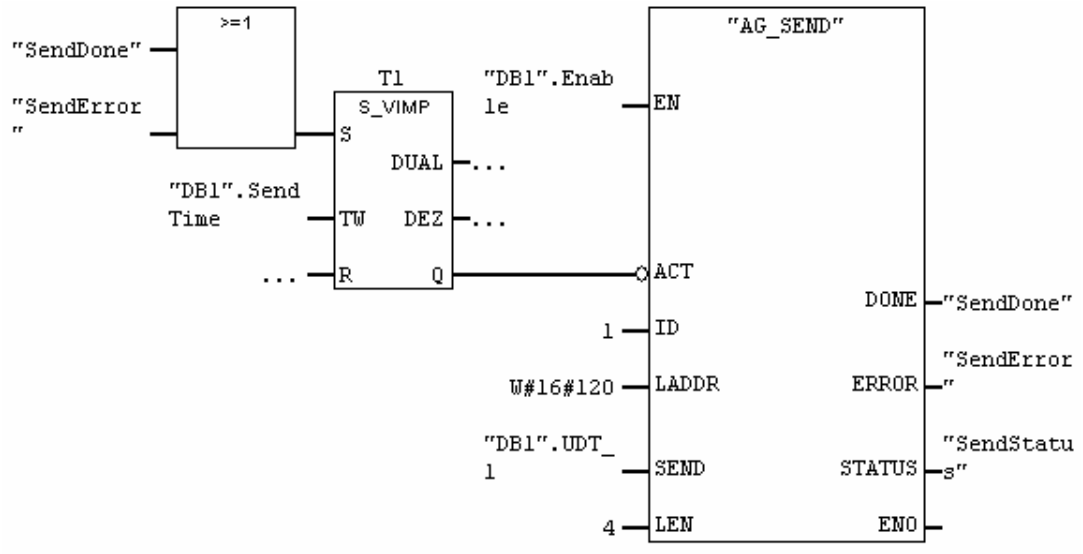


Bild 59: Funktionsplan zum Senden

So laden Sie den Code in die SIMATIC 300 Steuerung:

1. **Code Generator** für das Programm starten.
2. Sicherstellen, dass die Codes fehlerfrei generiert wurden.
3. Code in die SIMATIC 300 Steuerung laden.

8.2.2 S&R TCP Konfiguration der HIMax Steuerung

Für die Konfiguration der HIMax Steuerungen und den Umgang mit dem Programmiertool SILworX wird das Handbuch "Erste Schritte SILworX" empfohlen.

So erstellen Sie die folgenden Globalen Variablen im Variableneditor:

1. Im Strukturbaum **Konfiguration, Globale Variablen** wählen.
2. Rechtsklick auf **Globale Variablen** und **Edit** wählen.
3. Globale Variablen wie in der Tabelle 206 erstellen.

Name	Typ
Siemens_HIMA1	Byte
Siemens_HIMA2	Byte
Siemens_HIMA3	WORD
HIMA_Siemens1	Byte
HIMA_Siemens2	Byte
HIMA_Siemens3	WORD

Tabelle 206: Globale Variablen

So erstellen Sie das S&R-TCP-Protokoll in der Ressource:

1. Im Strukturbaum **Konfiguration**, **Ressource** öffnen.
2. Rechtsklick auf **Protokolle** und im Kontextmenü **Neu** wählen.
3. **Send/Receive over TCP** wählen und einen Namen für das Protokoll eintragen.
4. Mit OK bestätigen, um ein neues Protokoll zu erstellen.
5. Rechtsklick auf **Send/Receive over TCP** und im Kontextmenü **Eigenschaften** wählen.
6. **COM Modul** auswählen. Die restlichen Parameter behalten die Standardwerte.

So erstellen Sie die TCP Verbindung:

1. Rechtsklick auf **Send/Receive over TCP** und im Kontextmenü **Neu** wählen.
2. Rechtsklick auf **TCP Verbindung** und im Kontextmenü **Eigenschaften** wählen.
3. Eigenschaften wie im Bild unten Konfigurieren.

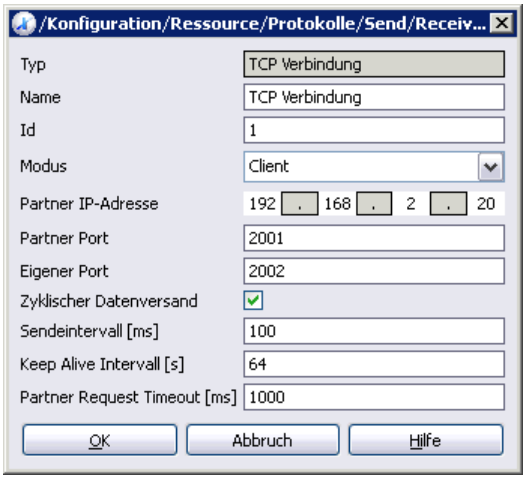


Bild 60: Eigenschaften der TCP-Verbindung in SILworX

i Wenn ein zyklischer Datenaustausch zwischen zwei Steuerungen parametriert werden soll, muss im Dialog *Eigenschaften* der TCP-Verbindung die Option *Zyklischer Datenversand* aktiviert sein.

So konfigurieren Sie die Empfangsdaten der HIMax Steuerung.

1. Rechtsklick auf **TCP-Verbindung** und im Kontextmenü **Edit** wählen.
2. Register **Prozessvariablen** wählen.
3. In der **Objektauswahl** die folgenden Globalen Variablen auswählen und diese per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Globale Variable	Typ
Siemens_HIMA1	Byte
Siemens_HIMA2	Byte
Siemens_HIMA3	WORD

Tabelle 207: Variablen für Empfangsdaten

4. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Eingänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

i Beachten Sie, dass die Offsets der Variablen in der HIMax Steuerung mit den Adressen der Variablen im Datentyp *UDT_1* der SIMATIC 300 übereinstimmen müssen.

So konfigurieren Sie die Sendedaten der HIMax Steuerung.

1. Rechtsklick auf **TCP-Verbindung** und im Kontextmenü **Edit** wählen.
2. Register **Prozessvariablen** wählen.
3. In der **Objektauswahl** die folgenden Globalen Variablen auswählen und diese per Drag&Drop in den Bereich **Eingangssignale** ziehen.

Globale Variable	Typ
HIMA_Siemens1	Byte
HIMA_Siemens2	Byte
HIMA_Siemens3	WORD

Tabelle 208: Variablen für Sendedaten

4. Kontextmenü durch Rechtsklick auf eine leere Stelle im Bereich **Register Eingänge** öffnen und **Neue Offsets** wählen, um die Offsets der Variablen neu zu nummerieren.

i Beachten Sie, dass die Offsets der Variablen in der HIMax Steuerung mit den Adressen der Variablen im Datentyp *UDT_1* der SIMATIC 300 übereinstimmen müssen.

S&R TCP Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, Send/Receive over TCP** öffnen.
2. Auf Schaltfläche **Verifikation** in Action Bar klicken und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

8.3 Menüfunktionen S&R-TCP-Protokoll

8.3.1 Edit

Das Dialogfenster **Edit** des S&R-TCP-Protokolls enthält das folgende Register.

Systemvariablen

Mit den *Systemvariablen* kann der Zustand vom TCP Send Receive Protokoll im Anwenderprogramm ausgewertet werden.

Element	Beschreibung
Anzahl aktive Verbindungen	Systemvariable, welche die Anzahl aktiver (ungestörter) Verbindungen liefert.
Anzahl gestörte Verbindungen	Systemvariable, welche die Anzahl gestörter Verbindungen liefert. Gestört bedeutet, dass die TCP Verbindung durch einen Timeout oder Fehler unterbrochen wurde.
Status	Keine Funktion

Tabelle 209: Systemvariablen S&R TCP

8.3.2 Eigenschaften

Der Datenaustausch über eine TCP-Verbindung erfolgt entweder zyklisch oder azyklisch. Für den azyklischen Datenaustausch werden die S&R-TCP-Funktionsbausteine benötigt.

Zyklischer und nicht zyklischer Datenaustausch können nicht zusammen auf einer Verbindung verwendet werden.

Allgemein

Name	Beschreibung
Typ	Send/Receive over TCP
Name	Name für das Send/Receive-over-TCP-Protokoll. Maximal 31 Zeichen.
Modul	Auswahl des COM-Moduls auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen. Deaktiviert: Kein Limit der CPU-Last, für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%

Tabelle 210: Allgemeine Eigenschaften S&R TCP

CPU/COM

Die vorgegebenen Parameter sorgen für den schnellstmöglichen Datenaustausch der S&R TCP-Daten zwischen dem COM-Modul (COM) und dem CPU-Modul (CPU) in der Steuerung. Diese Parameter sollten nur dann geändert werden, wenn eine Reduzierung der COM- und/oder CPU-Auslastung für eine Anwendung erforderlich sind und der Prozess dies zulässt.

i Die Änderung der Parameter wird nur dem erfahrenen Programmierer empfohlen. Eine Erhöhung der COM und CPU Aktualisierungszeit bedeutet auch, dass die tatsächliche Aktualisierungszeit der S&R TCP-Daten erhöht wird. Die Zeitanforderungen der Anlage sind zu prüfen.

Name	Beschreibung
Refresh Rate [ms]	Aktualisierungszeit in Millisekunden, mit der die Daten im S&R-TCP-Protokoll zwischen COM und CPU ausgetauscht werden. Ist die Refresh Rate Null oder kleiner als die Zykluszeit der Steuerung, dann erfolgt der Datenaustausch so schnell wie möglich. Wertebereich: 0 ... (2^{31}-1) Standardwert: 0
In einem Zyklus	Aktiviert: Transfer der S&R TCP-Daten von der CPU zur COM innerhalb eines Zyklus der CPU. Deaktiviert: Transfer der S&R TCP-Daten (maximal 1100 Byte pro Datenrichtung) von der CPU zur COM über mehrere Zyklen der CPU.

Tabelle 211: Parameter COM/CPU

8.4 Menüfunktionen TCP Verbindung

8.4.1 Edit

Über die Menüfunktion **Edit** erreichen Sie die Register Prozessvariablen und Systemvariablen.

Prozessvariablen

Eingangssignale

Die Variablen für den zyklischen Datenaustausch, die von dieser Steuerung empfangen werden sollen, werden im Bereich *Eingangssignale* eingetragen.

Im Register *Eingangssignale* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Variablen (Sendedaten) des Kommunikationspartners sein.

Ausgangssignale

Die Variablen für den zyklischen Datenaustausch, die von dieser Steuerung gesendet werden sollen, werden im Bereich *Ausgangssignale* eingetragen.

Im Register *Ausgangssignale* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Variablen (Empfangsdaten) des Kommunikationspartners sein.

8.4.2 Systemvariablen

Mit den Variablen im Register *Systemvariablen* kann der Zustand der TCP-Verbindung im Anwenderprogramm ausgewertet werden.

Name	Beschreibung
Bytes empfangen	Anzahl Bytes, die bisher empfangen wurden
Bytes versenden	Anzahl Bytes, die bisher gesendet wurden
Errorcode	Fehlercode der TCP-Verbindung. Siehe Kapitel 8.8.4.
Errorcode Zeitstempel [ms]	Millisekunden-Anteil des Zeitstempels. Zeitpunkt des Fehlereintritts.
Errorcode Zeitstempel [s]	Sekunden-Anteil des Zeitstempels. Zeitpunkt des Fehlereintritts.
Partner Request Timeout	Bei zyklischer Datenübertragung: Timeoutzeit, innerhalb der nach Datenversand mindestens eine Datensendung vom Kommunikationspartner empfangen werden muss. 0=Aus 1 ... 2^{32}-1 [ms]
Partner Verbindungszustand	Wird keine Datensendung innerhalb der Timeoutzeit empfangen, so wird der Status Partner Verbindungszustand auf *nicht verbunden* gesetzt und die Verbindung neu aufgesetzt. 0=Keine Verbindung 1=Verbindung OK
Status	Verbindungsstatus der TCP-Verbindung. (Siehe Kapitel 8.8.6)

Tabelle 212: Systemvariablen

8.4.3 Eigenschaften

Der Datenaustausch über eine TCP-Verbindung erfolgt entweder zyklisch oder azyklisch. Für den azyklischen Datenaustausch werden die S&R-TCP-Funktionsbausteine benötigt. Beim zyklischen Datenverkehr ist der Betrieb von S&R-TCP-Funktionsbausteinen nicht möglich.

Name	Beschreibung
Typ	TCP-Verbindung
Name	Beliebiger, eindeutiger Name für eine TCP-Verbindung. Maximal 31 Zeichen.
Id	Beliebige, aber eindeutige Identifikationsnummer ID für jede TCP-Verbindung. Die ID wird auch als Referenz in den S&R-TCP-Funktionsbausteinen benötigt. Wertebereich 0..255 Standardwert: 0
Modus	Server (Standardwert): Diese Station arbeitet als Server, d. h. im passiven Modus. Der Verbindungsaufbau muss durch den Kommunikationspartner (Client) initiiert werden. Nach dem ersten Verbindungsaufbau sind aber beide Stationen gleichberechtigt und können zu jedem Zeitpunkt Daten senden. Benötigt wird die Angabe des eigenen Ports.
	Server mit definiertem Partner: Diese Station arbeitet als Server, d.h. im passiven Modus. Der Verbindungsaufbau muss durch den Kommunikationspartner (Client) initiiert werden. Nach dem ersten Verbindungsaufbau sind aber beide Stationen gleichberechtigt und können zu jedem Zeitpunkt Daten senden. Wird hier die IP-Adresse und/oder Port des Kommunikationspartners eingetragen, dann kann nur der definierte Kommunikationspartner eine Verbindung aufnehmen. Alle anderen Stationen werden ignoriert. Wird einer der Parameter (IP-Adresse oder Port) auf Null gesetzt, findet für diesen Parameter keine Überprüfung statt.
	Client: Diese Station arbeitet als Client, d. h. die Station initiiert den Verbindungsaufbau mit dem Kommunikationspartner. Benötigt die Angabe von IP-Adresse und Port des Kommunikationspartners. Optional kann auch ein eigener Port angegeben werden.
Partner IP-Adresse	IP-Adresse des Kommunikationspartners. 0.0.0.0: beliebige IP-Adresse ist erlaubt. Gültiger Bereich: 1.0.0.0 ... 223.255.255.255, außer: 127.x.x.x Standardwert: 0
Partner Port	Port des Kommunikationspartners. Null: beliebiger Port. Reservierte oder bereits belegte Ports (1 ... 1024) werden von dem COM-BS abgelehnt. Wertebereich 0..65535 Standardwert: 0

Eigener Port	Eigener Port. Null bedeutet einen beliebigen Port. Reservierte oder bereits belegte Ports (1 ... 1024) werden vom COM-BS abgelehnt. Wertebereich 0..65535 Standardwert: 0
Zyklischer Datenversand	Deaktiviert (Standardwert): Zyklischer Datenversand ist deaktiviert. Der Datenaustausch über diese TCP-Verbindung muss mit Funktionsbausteinen programmiert werden. Es dürfen auf dieser Verbindung keine zyklischen E/A-Daten definiert sein.
	Aktiviert: Zyklischer Datenversand ist aktiv. Die Daten werden im Dialog Prozessvariablen der TCP-Verbindung definiert. Es müssen Empfangsdaten definiert sein. Es können auf dieser Verbindung keine Funktionsbausteine betrieben werden.
Sendeintervall [ms]	Nur editierbar bei zyklischem Datenversand. Hier wird das Sendeintervall eingestellt. Wertebereich 10 ... 2147483647 ms (Kleinere Werte werden auf 10 ms aufgerundet) Standardwert: 0
KeepAlive Intervall [s]	Ist die Zeit, bis die von TCP bereitgestellte Verbindungsüberwachung aktiv wird. Null deaktiviert die Verbindungsüberwachung. Werden innerhalb des eingestellten KeepAlive-Intervalls keine Daten ausgetauscht, werden KeepAlive-Proben an den Kommunikationspartner geschickt. Besteht die Verbindung noch, werden die KeepAlive-Proben vom Kommunikationspartner bestätigt. Findet in einer Zeit > 10 KeepAlive Intervall kein Datenaustausch zwischen den Partnern statt, wird die Verbindung geschlossen. Wird nach dem Senden eines Datenpakets keine Antwort empfangen so wird in bestimmten Intervallen das Datenpaket wiederholt. Die Verbindung wird nach 12 erfolglosen Wiederholungen abgebrochen (ca. 7 Minuten). Wertebereich 1... 65535s Standardwert: 0 = deaktiviert
Partner Request Timeout [ms]	Bei zyklischer Datenübertragung: Timeoutzeit, innerhalb der nach Datenversand mindestens eine Datensendung vom Kommunikationspartner empfangen werden muss. Wird keine Datensendung innerhalb der Timeoutzeit empfangen, so wird der *Partner Verbindungszustand* auf *nicht verbunden* gesetzt und die Verbindung neu aufgesetzt. Nach dem Schließen der Verbindung durch ein Timeout oder einen anderen Fehler baut die aktive Seite die Verbindung mit einer Verzögerung von 10 x PartnerRequestTimeout bzw. 10 Sekunden, wenn PartnerRequestTimeout = 0 ist, neu auf. Die passive Seite öffnet den Port bereits nach der halben Zeit. 0=Aus Wertebereich 1 ... 2^{32}-1 [ms] Standardwert: 0

Tabelle 213: Eigenschaften S&R TCP-Verbindung

8.5 Datenaustausch

S&R TCP arbeitet gemäß dem Client/Server Prinzip. Der Verbindungsaufbau muss durch den Kommunikationspartner initiiert werden, der als Client konfiguriert ist. Nach dem ersten Verbindungsaufbau sind aber beide Kommunikationspartner gleichberechtigt und können zu jedem Zeitpunkt Daten senden.

S&R TCP besitzt kein eigenes Protokoll zur Datensicherung, sondern benutzt dafür direkt das TCP/IP Protokoll. Da TCP die Daten in einem „Daten-Stream“ sendet, muss sichergestellt sein, dass die Offsets und die Typen der auszutauschenden Variablen auf der Empfangsseite und auf der Sendeseite identisch sind.

S&R TCP ist kompatibel zu der Siemens SEND/RECEIVE-Schnittstelle und erlaubt den zyklischen Datenaustausch mit den Siemens S7-Funktionsbausteinen AG_SEND (FC5) und AG_RECV (FC6) (siehe Kapitel 8.2, Beispiel S&R TCP Konfiguration).

Zudem stellt HIMA fünf S&R-TCP-Funktionsbausteine bereit, mit denen die Kommunikation über das Anwenderprogramm gesteuert und individuell angepasst werden kann. Mit den S&R-TCP-Funktionsbausteinen können beliebige Protokolle (z. B. Modbus), die über TCP übertragen werden, gesendet und empfangen werden.

8.5.1 TCP-Verbindungen

Für jede Verbindung über S&R TCP mit einem Kommunikationspartner muss mindestens eine TCP-Verbindung in der HIMax Steuerung erstellt werden.

In den *Eigenschaften* der TCP-Verbindung müssen die Identifikationsnummer der TCP-Verbindung und die Adressen/Ports der eigenen Steuerung und des Kommunikationspartners eingetragen werden.

Maximal 32 TCP-Verbindungen können in einer HIMax Steuerung erstellt werden.

Die erstellten TCP-Verbindungen müssen unterschiedliche Identifikationsnummern und unterschiedliche Adressen/Ports besitzen.

So erstellen Sie eine neue TCP-Verbindung:

1. Im Strukturbaum **Konfiguration**, **Ressource** öffnen.
2. Rechtsklick auf **Protokolle** und im Kontextmenü **Neu** wählen.
3. **Send/Receive over TCP** wählen und einen Namen für das Protokoll eintragen.
4. Mit OK bestätigen, um ein neues Protokoll zu erstellen.
5. Rechtsklick auf **Send/Receive over TCP** und im Kontextmenü **Eigenschaften** wählen.
6. **COM Modul** auswählen. Die restlichen Parameter behalten die Standardwerte.

TIPP Die HIMax Steuerung und das Fremdsystem müssen sich im gleichen Subnet befinden oder bei Verwendung eines Routers die entsprechenden Routingeinträge besitzen.

8.5.2 Zyklischer Datenaustausch

Wird zyklischer Datenaustausch verwendet, dann muss ein Sendeintervall in der HIMax Steuerung und im Kommunikationspartner festgelegt werden.

Das Sendeintervall legt das zyklische Intervall fest, innerhalb dessen der sendende Kommunikationspartner seine Variablen an den empfangenden Kommunikationspartner sendet.

- Um einen kontinuierlichen Datenaustausch zu gewährleisten, sollte bei beiden Kommunikationspartnern ungefähr das gleiche Sendeintervall festgelegt werden (siehe Kapitel 8.5.5, Flusskontrolle).
- Die Option *Zyklischer Datenversand* muss in der verwendeten TCP-Verbindung für den zyklischen Datenaustausch aktiviert sein.
- In einer TCP-Verbindung, in der die Option *zyklischer Datenversand* aktiviert ist, dürfen keine Funktionsbausteine verwendet werden.
- Die zu sendenden und zu empfangenden Variablen werden im Dialogfenster *Prozessvariablen* der TCP-Verbindung zugewiesen. Empfangsdaten **müssen** vorhanden sein, Sendedaten sind optional.

i Die gleichen Variablen (gleiche Offsets und Typen), die in der einen Station als Sendedaten definiert sind, müssen in der anderen Station als Empfangsdaten definiert werden.

8.5.3 Azyklischer Datenaustausch mit Funktionsbausteine

Der azyklische Datenaustausch wird in der HIMax Steuerung vom Anwenderprogramm über die S&R-TCP-Funktionsbausteine gesteuert.
Somit ist es möglich, mit einem Timer oder einen mechanischen Schalter an einem physikalischen Eingang der HIMax Steuerung, den Datenaustausch zu steuern.

- Die Option *Zyklischer Datenversand* muss in der verwendeten TCP-Verbindung deaktiviert sein.
- Zu einem Zeitpunkt darf immer nur ein TCP S/R Funktionsbaustein senden.
- Die zu sendenden oder zu empfangenden Variablen werden im Dialog *Prozessvariablen* der S&R-TCP-Funktionsbausteine (alle außer *Reset*) zugewiesen.

i Die gleichen Variablen (gleiche Offsets und Typen), die in der einen Station als Sendedaten definiert sind, müssen in der anderen Station als Empfangsdaten definiert werden.

8.5.4 Gleichzeitiger zyklischer und azyklischer Datenaustausch

Hierzu muss eine TCP-Verbindung für zyklische Daten und eine zweite für azyklische Daten konfiguriert werden. Beide TCP-Verbindungen müssen unterschiedliche *Partner IP-Adressen* und *Partner Ports* verwenden.

Eine einzelne TCP-Verbindung kann nicht für zyklischen und azyklischen Datenaustausch gemeinsam verwendet werden.

8.5.5 Flusskontrolle

Die Flusskontrolle ist ein Bestandteil von TCP und überwacht den kontinuierlichen Datenverkehr zwischen zwei Kommunikationspartnern.

Bei zyklischer Datenübertragung muss nach maximal 3 bis 5 gesendeten Paketen mindestens ein Paket empfangen werden, sonst blockiert das Senden, bis wieder ein Paket empfangen wird oder die Verbindungsüberwachung die Verbindung schließt.

Die Anzahl (3 ... 5) der möglichen Sendungen ohne Paketerhalt ist abhängig von der Größe der zu sendenden Pakete.

Anzahl=5 für kleine Pakete < 4kB.

Anzahl=3 für große Pakete ≥ 4kB.

- Bei der Projektierung ist darauf zu achten, dass keine der beiden Stationen mehr Daten sendet, als die andere synchron verarbeiten kann.
- Für den zyklischen Datenaustausch muss bei beiden Kommunikationspartnern ungefähr das gleiche Sendeintervall eingestellt werden.

8.6 Fremdsysteme mit Pad Bytes

Beim zyklischem und azyklischen Datenaustausch ist zu beachten, dass manche Steuerungen (z. B. SIMATIC 300) sogenannte *Pad Bytes* einfügen. Damit wird sichergestellt, dass alle Datentypen die größer als ein Byte sind, immer an einem geraden Offset beginnen und dass die Gesamtlänge der Pakete (in Byte) ebenfalls immer gerade ist.

In der HIMax Steuerung müssen für die *Pad Bytes*, *Dummy-Bytes* an den entsprechenden Stellen eingefügt werden.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	InOut_1	BYTE	B#16#0
+2.0	InOut_3	WORD	W#16#0
=4.0		END_STRUCT	

Bild 61: Variablenliste Siemens

In der Siemens Steuerung wird ein *Pad-Byte* (nicht sichtbar) eingefügt, damit die Variable *InOut_3* an einem geraden Offset beginnt.

Ausgangssignale

F	Name	Datentyp	ffse	Transfer-Operation	Globale Variable
1	InOut_1	BYTE	0	5	InOut_1
2	Dummy	BYTE	1	5	Dummy
3	InOut_3	WORD	2	5	InOut_3

Bild 62: Variablenliste HIMax

In der HIMax Steuerung muss ein *Dummy-Byte* eingefügt werden, damit die Variable *InOut_3* den gleichen Offset wie in der Siemens Steuerung hat.

8.7 S&R-TCP-Funktionsbausteine

Wenn die zyklische Datenübertragung zu unflexibel ist, können Daten auch mittels der S&R-TCP-Funktionsbausteine gesendet und empfangen werden. Die Option *Zyklischer Datenversand* muss in der verwendeten TCP-Verbindung deaktiviert werden.

Mit den S&R-TCP-Funktionsbausteinen kann der Anwender die Datenübertragung über TCP/IP den Erfordernissen seines Projekts optimal anpassen.

Die Funktionsbausteine werden im Anwenderprogramm parametriert. So können die Funktionen (Senden, Empfangen, Reset) der HIMax Steuerung im Anwenderprogramm gesetzt und ausgewertet werden.

S&R-TCP-Funktionsbausteine werden nur für den azyklischen Datenaustausch benötigt. Für den zyklischen Datenaustausch zwischen Server und Client sind diese Funktionsbausteine nicht erforderlich!

i Die Konfiguration der S&R-TCP-Funktionsbausteine wird im Kapitel 12.1 beschrieben.

Es stehen die folgenden Funktionsbausteine zur Verfügung:

Funktionsbaustein	Beschreibung der Funktion
TCP_Reset (siehe Kapitel 8.7.1)	Rücksetzen einer TCP-Verbindung
TCP_Send (siehe Kapitel 8.7.2)	Senden von Daten
TCP_Receive (siehe Kapitel 8.7.3)	Empfangen von Datenpaketen fester Länge
TCP_ReceiveLine (siehe Kapitel 8.7.4)	Empfang einer ASCII-Zeile
TCP_ReceiveVar (siehe Kapitel 8.7.5)	Empfangen von Datenpaketen variabler Länge (mit Längenfeld)
LATCH	Wird nur innerhalb anderer Funktionsbausteine verwendet
PIG	Wird nur innerhalb anderer Funktionsbausteine verwendet
PIGII	Wird nur innerhalb anderer Funktionsbausteine verwendet

Tabelle 214: Funktionsbausteine für S&R TCP Verbindungen

8.7.1 TCP_Reset

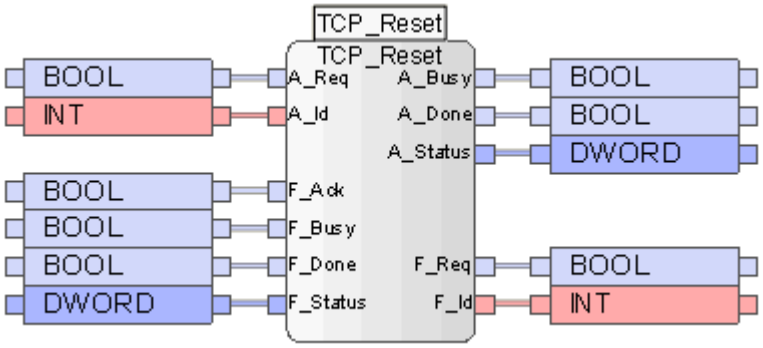


Bild 63: Funktionsbaustein TCP_Reset

Mit dem Funktionsbaustein **TCP Reset** kann eine gestörte Verbindung wiederhergestellt werden, wenn sich ein Send- oder Receive-Funktionsbaustein mit einem TIMEOUT-Fehler meldet (16#8A).

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Baustein	BOOL
A_Id	Identifikationsnummer *ID* der gestörten TCP-Verbindung, die zurückgesetzt werden soll.	INT

Tabelle 215: A-Eingänge Funkionsbaustein TCP_Reset

A-Ausgänge	Beschreibung	Typ
A_Busy	TRUE: Der Reset des Funktionsbausteins ist noch nicht beendet.	BOOL
A_Done	TRUE: Der Sendevorgang wurde fehlerfrei beendet	BOOL
A_Status	Am Ausgang *A_Status* wird Status und Fehlercode des Funktionsbausteins und der TCP-Verbindung ausgegeben.	DWORD

Tabelle 216: A-Ausgänge Funktionsbaustein TCP_Reset

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **Reset** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **Reset** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **TCP_Reset** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **TCP_Reset** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **Reset** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Busy	BOOL
F_Done	BOOL
F_Status	DWORD

Tabelle 217: F-Eingänge Funktionsbaustein TCP_Reset

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **TCP_Reset** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **Reset** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD

Tabelle 218: F-Ausgänge Funktionsbaustein TCP_Reset

Zugehörigen Funktionsbaustein Reset im Strukturbaum erstellen:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **Send Receive over TCP**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **Reset** wählen.
3. Rechtsklick auf Funktionsbaustein **Reset** und **Edit** wählen
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **Reset** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **TCP_Reset** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
REQ	BOOL

Tabelle 219: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **Reset** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **TCP_Reset** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
DONE	BOOL
STATUS	DWORD

Tabelle 220: Ausgangs-Systemvariablen

Für die Bedienung des Funktionsbaustein TCP_Reset sind die folgenden Schritte erforderlich:

1. Im Anwenderprogramm am Eingang *A_ID* die Identifikationsnummer der gestörten TCP-Verbindung setzen.
2. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis ein Reset an die definierte TCP-Verbindung gesendet wurde. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Done* auf TRUE.

8.7.2 TCP_Send

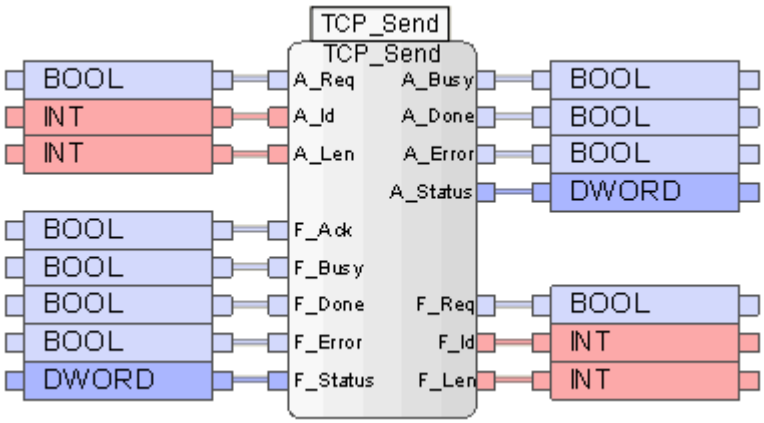


Bild 64: Funktionsbaustein TCP_Send

Der Funktionsbaustein **TCP_Send** dient zum azyklischen Senden von Variablen zu einem Kommunikationspartner. Im Kommunikationspartner muss ein Funktionsbaustein z. B. *Empfangen* mit den gleichen Variablen und Offsets konfiguriert werden.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Baustein.	BOOL
A_Id	Identifikationsnummer der konfigurierten TCP-Verbindung zu dem Kommunikationspartner, zu welchem die Daten gesendet werden sollen.	INT
A_Len	Anzahl der zu sendenden Variablen in Bytes. A_Len muss größer als Null sein und darf nicht innerhalb einer Variablen enden.	INT

Tabelle 221: A-Eingänge Funktionsbaustein TCP_Send

A-Ausgänge	Beschreibung	Typ
A_Busy	TRUE: Der Sendevorgang ist noch nicht beendet.	BOOL
A_Done	TRUE: Der Sendevorgang wurde fehlerfrei beendet	BOOL
A_Error	TRUE: Ein Fehler ist aufgetreten FALSE: Kein Fehler	BOOL
A_Status	Am Ausgang *A_Status* wird Status und Fehlercode des Funktionsbausteins und der TCP-Verbindung ausgegeben.	DWORD

Tabelle 222: A-Ausgänge Funktionsbaustein TCP_Send

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **Senden** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **Senden** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **TCP_Send** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **TCP_Send** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **Senden** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Busy	BOOL
F_Done	BOOL
F_Error	BOOL
F_Status	DWORD

Tabelle 223: F-Eingänge Funktionsbaustein TCP_Send

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **TCP_Send** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **Senden** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Id	DWORD
F_Len	INT
F_Req	BOOL

Tabelle 224: F-Ausgänge Funktionsbaustein TCP_Send

Zugehörigen Funktionsbaustein Senden im Strukturbaum erstellen:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **Send Receive over TCP**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **Senden** wählen.
3. Rechtsklick auf Funktionsbaustein **Senden** und **Edit** wählen
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **Senden** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **TCP_Send** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	DWORD
LEN	INT
REQ	BOOL

Tabelle 225: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **Senden** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **TCP_Send** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
Ack	BOOL
Busy	BOOL
Done	BOOL
ERROR	BOOL
STATUS	DWORD

Tabelle 226: Ausgangs-Systemvariablen

Daten	Beschreibung
Sende-Daten	Im Register *Prozessvariablen* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Variablen des Kommunikationspartners sein.

Tabelle 227: Sende-Daten

Für die Bedienung des Funktionsbaustein TCP_Send sind die folgenden Schritte erforderlich:

i Die zu sendenden Variablen müssen im Register *Prozessvariablen* des Dialogs *Senden* angelegt werden. Die Offsets und Typen der Variablen müssen identisch mit den Offsets und den Typen der Variablen des Kommunikationspartners sein.

1. Im Anwenderprogramm die TCP-Verbindungs ID am Eingang *A_ID* setzen.
2. Im Anwenderprogramm die Länge der zu sendenden Variablen am Eingang *A_Len* in Byte setzen.
3. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang A_Busy geht solange auf TRUE, bis die Variablen gesendet wurden. Danach gehen die Ausgänge A_Busy auf FALSE und A_Done auf TRUE.

Konnte der Sendevorgang nicht erfolgreich ausgeführt werden, geht der Ausgang A_Error auf TRUE und am Ausgang A_Status wird ein Fehlercode ausgegeben.

8.7.3 TCP_Receive

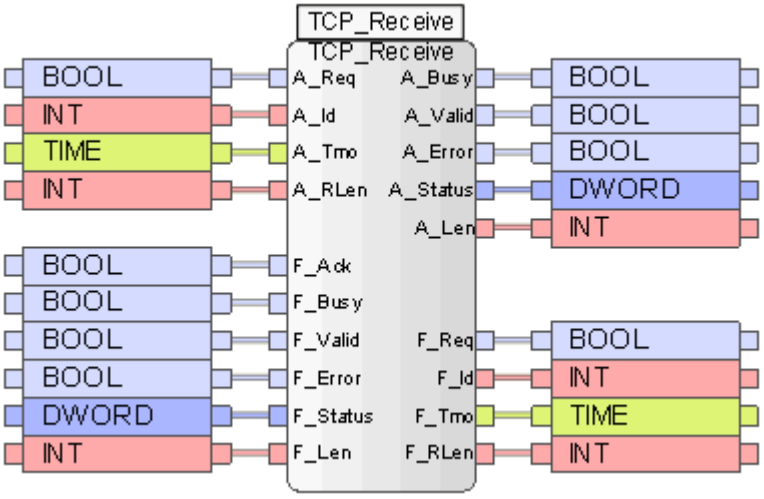


Bild 65: Funktionsbaustein TCP_Receive

Mit dem Funktionsbaustein **TCP_Receive** können definierte Variablen von einem Kommunikationspartner empfangen werden.

Im Kommunikationspartner muss ein Funktionsbaustein z. B. TCP_Send mit den gleichen Variablen und Offsets konfiguriert werden.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Funktionsbaustein.	BOOL
A_Id	Identifikationsnummer der konfigurierten TCP-Verbindung zu dem Kommunikationspartner, von welchem die Daten empfangen werden sollen.	INT
A_Tmo	Empfangs-Timeout. Wenn innerhalb dieser Zeit keine Daten empfangen wurden, wird der Baustein mit einer Fehlermeldung beendet. Wird der Eingang *A_Tmo* nicht belegt, oder null angelegt, ist der Timeout deaktiviert.	TIME
A_RLen	*A_RLen* ist die erwartete Länge der zu empfangenden Variablen in Bytes. *A_RLen* muss größer als null sein und darf nicht innerhalb einer Variablen enden.	INT

Tabelle 228: A-Eingänge Funktionsbaustein TCP_Receive

A-Ausgänge	Beschreibung	Typ
A_Busy	TRUE: Der Empfang der Daten ist noch nicht beendet.	BOOL
A_Valid	TRUE: Der Empfang der Daten wurde fehlerfrei beendet.	BOOL
A_Error	TRUE: Ein Fehler ist aufgetreten FALSE: Kein Fehler	BOOL
A_Status	Am Ausgang *A_Status* werden Status und Fehlercode des Funktionsbausteins und der TCP-Verbindung ausgegeben.	DWORD
A_Len	Anzahl der empfangenen Bytes.	INT

Tabelle 229: A-Ausgänge Funktionsbaustein TCP_Receive

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **Empfangen** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **Empfangen** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **TCP_Receive** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **TCP_Receive** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **Empfangen** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
F_Status	DWORD
F_Len	INT

Tabelle 230: A-Eingänge Funktionsbaustein TCP_Receive

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **TCP_Receive** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **Empfangen** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	DWORD
F_Tmo	INT
F_RLen	INT

Tabelle 231: F-Ausgänge Funktionsbaustein TCP_Receive

Zugehörigen Funktionsbaustein Empfangen im Strukturbaum erstellen:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **Send Receive over TCP**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **Empfangen** wählen.
3. Rechtsklick auf Funktionsbaustein **Empfangen** und **Edit** wählen
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **TCP_Receive** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	INT
REQ	BOOL
RLEN	INT
TIMEOUT	TIME

Tabelle 232: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **TCP_Receive** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
Ack	BOOL
Busy	BOOL
ERROR	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Tabelle 233: Ausgangs-Systemvariablen

Daten	Beschreibung
Empfangsvariablen	Im Register *Prozessvariablen* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Variablen des Kommunikationspartners sein.

Tabelle 234: Empfangsvariablen

Für die Bedienung des Funktionsbaustein TCP_Receive sind die folgenden Schritte erforderlich:

i Die Empfangsvariablen müssen im Register *Prozessvariablen* des Dialogs *Empfangen* angelegt werden. Die Offsets und Typen der Empfangsvariablen müssen identisch mit den Offsets und den Typen der Sendevariablen des Kommunikationspartners sein.

1. Im Anwenderprogramm die Identifikationsnummer der TCP-Verbindung am Eingang *A_ID* setzen.
2. Im Anwenderprogramm die Empfangs-Timeout am Eingang *A_Tmo* setzen.
3. Im Anwenderprogramm die erwartete Länge der zu empfangenden Variablen am Eingang *A_RLen* setzen.
4. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein startet mit einem positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis die Variablen empfangen wurden, oder der Empfangs-Timeout abgelaufen ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Valid* oder *A_Error* auf TRUE.

Ist der Empfang der Variablen fehlerfrei, wechselt der Ausgang *A_Valid* auf TRUE. Die Variablen, die im Register *Daten* definiert wurden, können ausgewertet werden.

Ist der Empfang der Variablen nicht fehlerfrei, wechselt der Ausgang *A_Error* auf TRUE, und am Ausgang *A_Status* wird ein Fehlercode ausgegeben.

8.7.4 TCP_ReceiveLine

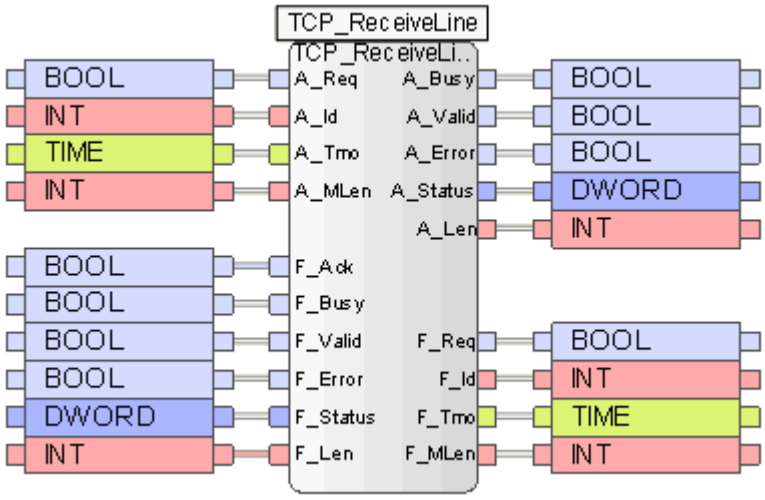


Bild 66: Funktionsbaustein TCP_ReceiveLine

Der Funktionsbaustein **TCP_ReceiveLine** dient zum Empfang einer ASCII Zeichenkette inklusive LineFeed (16#0A) eines Kommunikationspartners.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A:

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Positive Flanke startet den Baustein und setzt die Verbindung empfangsbereit	BOOL
A_Id	Identifikationsnummer der konfigurierten TCP-Verbindung zu dem Kommunikationspartner, von welchem die Daten empfangen werden sollen.	INT
A_Tmo	Empfangs-Timeout Wenn innerhalb dieser Zeit keine Daten empfangen wurden, wird der Baustein mit einer Fehlermeldung beendet. Wird der Eingang offen gelassen, oder Null angelegt, ist der Timeout ausgeschaltet.	TIME
A_MLen	*A_Mlen* ist die maximale Länge einer zu empfangenden Zeile in Bytes. Die Empfangsvariablen müssen im Register *Daten* im Com-Funktionsbaustein angelegt werden. Übertragene Bytes = Min (A_MLen, Zeilenlänge, Länge des Datenbereichs).	INT

Tabelle 235: A-Eingänge Funktionsbaustein TCP_ReceiveLine

A-Ausgänge	Beschreibung	Typ
A_Busy	TRUE: Der Empfang der Daten ist noch nicht beendet.	BOOL
A_Valid	TRUE: Der Empfang der Daten wurde fehlerfrei beendet.	BOOL
A_Error	TRUE: Ein Fehler ist aufgetreten FALSE: Kein Fehler	BOOL
A_Status	Am Ausgang A_Status wird Status und Fehlercode des Funktionsbausteins und der TCP-Verbindung ausgegeben.	DWORD
A_Len	Anzahl der empfangenen Bytes.	INT

Tabelle 236: A-Ausgänge Funktionsbaustein TCP_ReceiveLine

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **Zeilenweises Empfangen** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **Zeilenweises Empfangen** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **TCP_ReceiveLine** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **TCP_ReceiveLine** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **Zeilenweises Empfangen** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
F_Status	DWORD
F_Len	INT

Tabelle 237: F-Eingänge Funktionsbaustein TCP_ReceiveLine

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **TCP_ReceiveLine** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **Zeilenweises Empfangen** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
A_Req	BOOL
A_Id	INT
A_Tmo	TIME
A_MLen	INT

Tabelle 238: F-Ausgänge Funktionsbaustein TCP_ReceiveLine

Zugehörigen Funktionsbaustein Zeilenweises Empfangen im Strukturbaum erstellen:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **Send Receive over TCP**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **Zeilenweises Empfangen** wählen.
3. Rechtsklick auf Funktionsbaustein **Zeilenweises Empfangen** und **Edit** wählen
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **Zeilenweises Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **TCP_ReceiveLine** im Anwenderprogramm verbunden haben

Eingänge	Typ
ID	INT
MLEN	INT
REQ	BOOL
TIMEOUT	TIME

Tabelle 239: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **Zeilenweises Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **TCP_ReceiveLine** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
ERROR	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Tabelle 240: Ausgangs-Systemvariablen

Daten	Beschreibung
Empfangsvariablen	Im Register *Prozessvariablen* sollten sinnvollerweise nur Variablen vom Typ BYTE angelegt werden. Die Offsets der Variablen müssen identisch mit den Offsets der Variablen des Kommunikationspartners sein.

Tabelle 241: Empfangsvariablen

Für die Bedienung des Funktionsbaustein TCP_ReceiveLine sind die folgenden Schritte erforderlich:

i Die Empfangsvariablen vom Typ Byte müssen im Register *Prozessvariablen* des Dialogs *Zeilenweises Empfangen* angelegt werden. Die Offsets der Empfangsvariablen müssen identisch mit den Offsets der Sendevariablen des Kommunikationspartners sein.

1. Im Anwenderprogramm die Identifikationsnummer der TCP-Verbindung am Eingang *A_ID* setzen.
2. Im Anwenderprogramm die Empfangs-Timeout am Eingang *A_Tmo* setzen.
3. Im Anwenderprogramm die maximale Länge der zu empfangenden Zeile am Eingang *A_MLen* setzen.

i *A_MLen* muss größer als Null sein und bestimmt die Größe des Empfangspuffers in Byte.

Wenn der Empfangspuffer gefüllt ist, und noch kein Zeilenende aufgetreten ist, wird der Lesevorgang ohne Fehlermeldung beendet.

Am Ausgang *A_Len* wird die Anzahl der empfangenen Bytes zur Verfügung gestellt:

Empfangene Bytes = Min (A_MLen, Zeilenlänge, Länge des Datenbereichs).

4. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein reagiert auf einen positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis der Empfangspuffer voll ist oder das Zeilenende *LineFeed* empfangen wurde oder der Empfangs-Timeout abgelaufen ist. Danach gehen die Ausgänge *A_Busy* auf FALSE und *A_Valid* oder *A_Error* auf TRUE.

Ist der Empfang der Zeile fehlerfrei, wechselt der Ausgang *A_Valid* auf TRUE. Die Variablen, die im Register Daten definiert wurden, können ausgewertet werden.

Ist der Empfang der Zeile nicht fehlerfrei, wechselt der Ausgang *A_Error* auf TRUE, und am Ausgang *A_Status* wird ein Fehlercode ausgegeben.

8.7.5 TCP_ReceiveVar

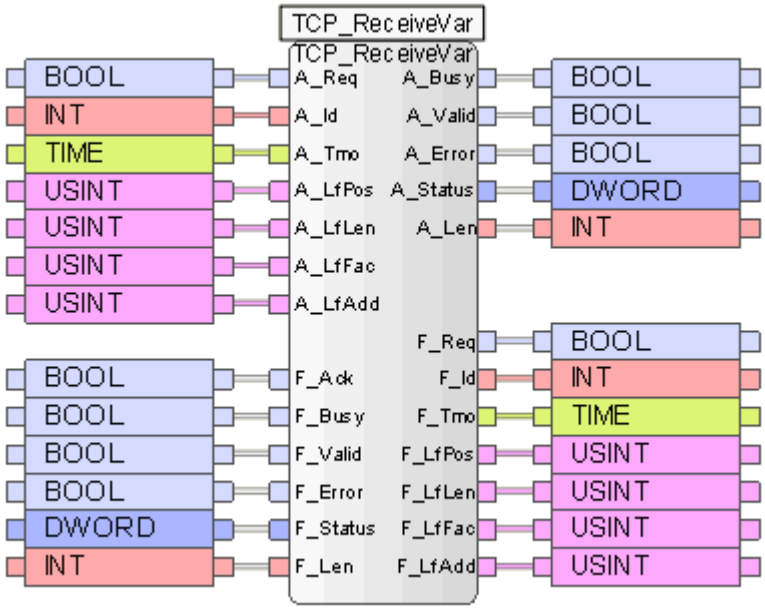


Bild 67: Funktionsbaustein TCP_ReceiveVar

Mit dem Funktionsbaustein **TCP_ReceiveVar** können Datenpakete variabler Länge, die mit einem Längenfeld ausgestattet sind, ausgewertet werden.

i Zur Konfiguration ziehen Sie den Funktionsbaustein per Drag&Drop aus der Bausteinbibliothek in das Anwenderprogramm (siehe auch Kapitel 12.1).

Funktionsbeschreibung

Die empfangenen Datenpakete müssen den im Bild unten dargestellten Aufbau besitzen (z. B. Modbus-Protokoll). Eine Anpassung an ein beliebiges Protokoll-Format erfolgt über die Einstellung der Eingabeparameter *A_LfPos*, *A_LfLen*, *A_LfFac*, *A_LfLen*.

Das empfangene Datenpaket besteht aus einem Kopf- und einem Nutzdatenbereich. Der Kopfbereich enthält Daten wie Teilnehmer-Adresse, Telegrammfunktion, Längenfeld usw., die für die Kommunikationsverbindung erforderlich sind. Um den Nutzdatenbereich auszuwerten muss der Kopfbereich abgetrennt und das Längenfeld ausgelesen werden.

Die Größe des Kopfbereichs wird im Parameter *A_LfAdd* eingetragen.

Die Länge des Nutzdatenbereichs muss aus dem Längenfeld des aktuell gelesenen Datenpakets ausgelesen werden. Die Position des Längenfeldes wird im Parameter *A_LfPos* eingetragen. Die Größe des Längenfeldes wird in *LfLen* in Byte eingetragen. Falls die Länge nicht als Byte angegeben ist, muss der Umrechnungsfaktor hierfür in *A_LfFac* eingetragen werden (z. B. 2 für Word oder 4 für Double Word).

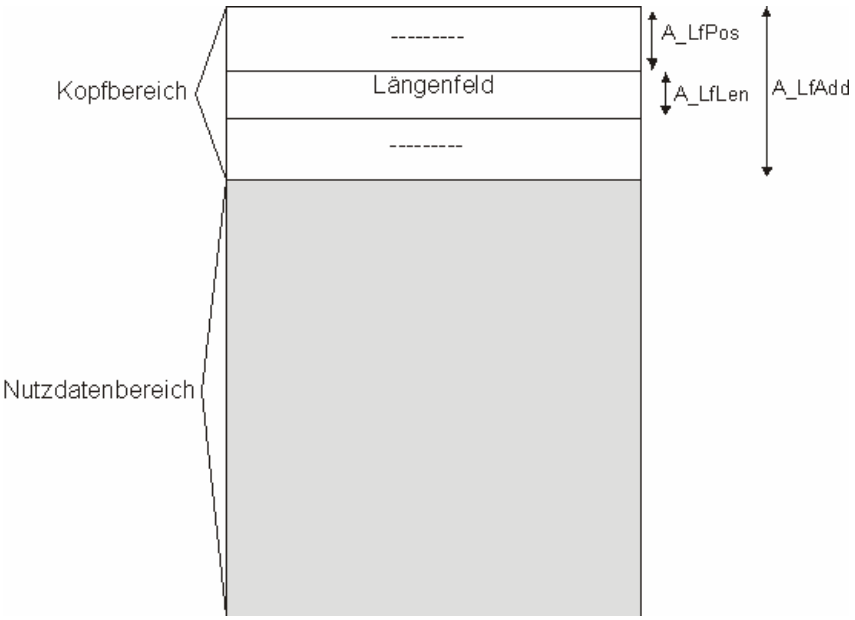


Bild 68: Aufbau der Datenpakete

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix A

Über diese Ein- und Ausgänge können Sie den Funktionsbaustein mit Hilfe des Anwenderprogramms steuern und auswerten. Das Präfix "A" steht für "Application".

A-Eingänge	Beschreibung	Typ
A_Req	Mit der positiven Flanke wird der CPU-Funktionsbaustein gestartet	BOOL
A_Id	Identifikationsnummer *ID* der konfigurierten TCP-Verbindung zu einem Kommunikationspartner, von dem das Datenpaket empfangen werden soll.	DWORD
A_Tmo	Empfangs-Timeout Wenn innerhalb dieser Zeit keine Daten empfangen wurden, wird der Baustein mit einer Fehlermeldung beendet. Wird der Eingang offen gelassen, oder null angelegt, ist der Timeout ausgeschaltet.	INT
A_LfPos	Startposition des Längenfeldes im Datenpaket; die Nummerierung beginnt mit null (gemessen in Bytes).	USINT
A_LfLen	Größe des Längenfelds *A_LfLen* in Bytes. Erlaubt sind 1, 2 oder 4 Bytes.	USINT
A_LfFac	Umrechnungsfaktor in Bytes, falls der Eintrag im Längenfeld nicht in Bytes ist. Wird der Eingang offen gelassen, oder mit Null belegt, wird **1** als Standardwert genommen.	USINT
A_LfAdd	Größe des Kopffeldes in Bytes	USINT

Tabelle 242: A-Eingänge Funktionsbaustein TCP_ReceiveVar

A-Ausgänge	Beschreibung	Typ
A_Busy	TRUE: Der Empfang der Daten ist noch nicht beendet.	BOOL
A_Valid	TRUE: Der Empfang der Daten wurde Fehlerfrei beendet.	BOOL
A_Error	TRUE: Beim Lesen trat ein Fehler auf FALSE: Kein Fehler	BOOL
A_Status	Am Ausgang *A_Status* wird Status und Fehlercode des Funktionsbausteins und der TCP-Verbindung ausgegeben.	DWORD
A_Len	Anzahl der empfangenen Bytes.	INT

Tabelle 243: A-Ausgänge Funktionsbaustein TCP_ReceiveVar

Ein- und Ausgänge des Funktionsbausteins mit dem Präfix F:

Diese Ein- und Ausgänge des Funktionsbausteins stellen die Verbindung zum Funktionsbaustein **Variabel Empfangen** im Strukturbaum her. Das Präfix "F" steht für "Field".

i Die Verbindung des Funktionsbausteins **Variabel Empfangen** im Strukturbaum (im Ordner Funktionsbausteine) mit dem Funktionsbaustein **TCP_ReceiveVar** (im Anwenderprogramm) erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

Verbinden Sie die *F-Eingänge* des Funktionsbausteins **TCP_ReceiveVar** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Ausgänge des Funktionsbausteins **Variabel Empfangen** im Strukturbaum verbinden werden.

F-Eingänge	Typ
F_Ack	BOOL
F_Busy	BOOL
F_Valid	BOOL
F_Error	BOOL
A_Status	DWORD
A_Len	INT

Tabelle 244: F-Eingänge Funktionsbaustein TCP_ReceiveVar

Verbinden Sie die *F-Ausgänge* des Funktionsbausteins **TCP_ReceiveVar** im Anwenderprogramm mit den gleichen Variablen, mit denen Sie später auch die Eingänge des Funktionsbausteins **Variabel Empfangen** im Strukturbaum verbinden werden.

F-Ausgänge	Typ
F_Req	BOOL
F_Id	INT
F_Tmo	TIME
F_LfPos	USINT
A_LfLen	USINT
A_LfFac	USINT
A_LfAdd	USINT

Tabelle 245: F-Ausgänge Funktionsbaustein TCP_ReceiveVar

Zugehörigen Funktionsbaustein Variabel Empfangen im Strukturbaum erstellen:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle**, **Send Receive over TCP**, **Funktionsbausteine**, **Neu** öffnen.
2. Funktionsbaustein **Variabel Empfangen** wählen.
3. Rechtsklick auf Funktionsbaustein **Variabel Empfangen** und **Edit** wählen
   ☑ Variablenzuweisung zum Funktionsbaustein wird geöffnet.

Verbinden Sie die Eingänge des Funktionsbausteins **Variabel Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Ausgänge* des Funktionsbausteins **TCP_ReceiveVar** im Anwenderprogramm verbunden haben.

Eingänge	Typ
ID	INT
Lf Add	USINT
Lf Fac	USINT
Lf Len	USINT
Lf Pos	USINT
REQ	BOOL
TIMEOUT	TIME

Tabelle 246: Eingangs-Systemvariablen

Verbinden Sie die folgenden Ausgänge des Funktionsbausteins **Variabel Empfangen** im Strukturbaum mit den gleichen Variablen, mit denen Sie zuvor auch die *F-Eingänge* des Funktionsbausteins **TCP_ReceiveVar** im Anwenderprogramm verbunden haben.

Ausgänge	Typ
ACK	BOOL
BUSY	BOOL
ERROR	BOOL
LEN	INT
STATUS	DWORD
VALID	BOOL

Tabelle 247: Ausgangs-Systemvariablen

Daten	Beschreibung
Empfangsvariablen	Im Register *Prozessvariablen* können beliebige Variablen angelegt werden. Die Offsets und Typen der Variablen müssen allerdings identisch mit den Offsets und den Typen der Variablen des Kommunikationspartners sein.

Tabelle 248: Empfangsvariablen

Für die Bedienung des Funktionsbaustein TCP_ReceiveVar sind die folgenden Schritte erforderlich:

i Die Empfangsvariablen müssen im Register *Prozessvariablen* des Dialogs *Variabel Empfangen* angelegt werden. Die Offsets und Typen der Empfangsvariablen müssen identisch mit den Offsets und den Typen der Sendevariablen des Kommunikationspartners sein.

1. Im Anwenderprogramm die Identifikationsnummer der TCP-Verbindung am Eingang *A_ID* setzen.
2. Im Anwenderprogramm die Empfangs-Timeout am Eingang *A_Tmo* setzen.
3. Im Anwenderprogramm die Parameter *A_LfPos*, *A_LfLen*, *A_LfFac* und *A_LfAdd* setzen.
4. Im Anwenderprogramm den Eingang *A_Req* auf TRUE setzen.

i Der Funktionsbaustein startet mit einem positiven Flankenwechsel an *A_Req*.

Der Ausgang *A_Busy* ist TRUE, bis die Variablen empfangen wurden, oder der Empfangs-Timeout abgelaufen ist. Danach wechseln die Ausgänge *A_Busy* auf FALSE und *A_Valid* oder *A_Error* auf TRUE.

Ist der Empfang der Variablen fehlerfrei, wechselt der Ausgang *A_Valid* auf TRUE. Die Variablen, die im Register Daten definiert wurden, können ausgewertet werden. Der Ausgang *A_Len* enthält die Anzahl der Bytes, die tatsächlich ausgelesen wurden.

Ist der Empfang der Variablen nicht fehlerfrei, wechselt der Ausgang *A_Error* auf TRUE, und am Ausgang *A_Status* wird ein Fehlercode ausgegeben.

8.8 Control-Panel (Send/Receive over TCP)

Im Control-Panel kann der Anwender die Einstellungen des Send/Receive Protokolls überprüfen und steuern. Zudem werden aktuelle Statusinformationen (z. B. gestörte Verbindungen usw.) des Send/Receive Protokolls angezeigt.

So öffnen Sie das Control Panel zur Überwachung des Send/Receive Protokolls:

1. Im Strukturbaum **Ressource** wählen.
2. Aus der **Action Bar** Online wählen.
3. Im **System-Login**, Zugangsdaten eingeben um das Control Panel der Ressource zu öffnen.
4. Im Strukturbaum des Control Panels **Send/Receive Protokoll** wählen.

8.8.1 Kontextmenü (Send/Receive Protokoll)

Aus dem Kontextmenü des selektierte Send/Receive Protokolls können Sie das folgende Kommando wählen:

Reset:

Setzt die statistischen Daten (Zykluszeit min, max usw.) auf null zurück.

8.8.2 Anzeigefeld (Send/Receive Protokoll)

In dem Anzeigefeld werden die folgenden Werte des selektierte Send/Receive Protokolls angezeigt.

Element	Beschreibung
Name	TCP SR Protokoll
CPU-Last (projektierte) [%]	Siehe Kapitel 7.4.2.
CPU-Last (tatsächliche) [%]	
Ungestörte Verbindungen	Anzahl gestörte Verbindungen
Gestörte Verbindungen	Anzahl gestörte Verbindungen

Tabelle 249: Anzeigefeld S&R Protokoll

8.8.3 Anzeigefeld (Send/Receive Server)

In dem Anzeigefeld werden die folgenden Werte des selektierten Modbus Clients angezeigt.

Element	Beschreibung
Name	Name Modbus Slaves
Partner Timeout [ms]	Timeoutzeit, innerhalb der nach Datenversand mindestens eine Datensendung vom Kommunikationspartner empfangen wurde.
Verbindungszustand	Aktueller Verbindungszustand dieser Verbindung 0x00: Verbindung OK 0x01: Verbindung geschlossen 0x02: Server wartet auf Verbindungsaufnahme 0x04: Client versucht Verbindungsaufbau 0x08: Verbindung ist blockiert
Peer-Adresse	IP-Adresse des Kommunikationspartners.
Peer-Port	Port des Kommunikationspartners.
Eigener Port	Port dieser Steuerung.
Fehlercode	Fehlercode (siehe Kapitel 8.8.4)

Tabelle 250: Anzeigefeld Modbus Slaves

8.8.4 Fehlercode der TCP-Verbindung

Die Fehlercodes können aus der Variablen *Errorcode* gelesen werden.

Pro konfigurierter Verbindung: Der Verbindungsstatus setzt sich zusammen aus dem Verbindungszustand und dem Fehlercode der letzten Operation.

Fehlercode Dezimal	Fehlercode Hexadezimal	Beschreibung
0	16#00	OK
4	16#04	Unterbrochener Systemaufruf
5	16#05	I/O-Fehler
6	16#06	Unbekanntes Gerät
9	16#09	Ungültiger Socket-Deskriptor
12	16#0C	Kein Speicher mehr vorhanden
13	16#0D	Zugriff verweigert
14	16#0E	Ungültige Adresse
16	16#10	Gerät ist beschäftigt
22	16#16	Ungültiger Wert (z. B. im Längenfeld)
23	16#17	Deskriptortabelle ist voll
32	16#20	Verbindung unterbrochen
35	16#23	Operation ist blockiert
36	16#24	Operation jetzt in Arbeit
37	16#25	Operation bereits in Arbeit
38	16#27	Zieladresse erforderlich
39	16#28	Message zu lang
40	16#29	Falscher Protokolltyp für Socket
42	16#2A	Protokoll nicht verfügbar
43	16#2B	Protokoll nicht unterstützt
45	16#2D	Operation auf Socket nicht unterstützt
47	16#2F	Adresse von Protokoll nicht unterstützt
48	16#30	Adresse ist bereits in Verwendung
49	16#31	Adresse kann nicht zugewiesen werden
50	16#32	Netzwerk läuft nicht
53	16#35	Software hat Verbindung abgebrochen
54	16#36	Verbindung wurde vom Partner zurückgesetzt
55	16#37	Kein Pufferspeicher mehr verfügbar
56	16#38	Socket ist bereits verbunden
57	16#39	Socket ist nicht verbunden
58	16#3A	Socket ist geschlossen
60	16#3C	Zeit für Operation abgelaufen
61	16#3D	Verbindung abgewiesen (durch Partner)
65	16#41	Kein Routingeintrag zum Partner vorhanden
78	16#4E	Funktion nicht vorhanden
254	16#FE	Timeout aufgetreten
255	16#FF	Verbindung vom Partner geschlossen

Tabelle 251: Fehlercodes der TCP-Verbindung

8.8.5 Zusätzliche Fehlercodetabelle der Funktionsbausteine

Die Fehlercodes der Funktionsbausteine (z. B. 16#8x) werden nur an A_Status der S&R-TCP-Funktionsbausteinen ausgegeben.

Fehlercode Dezimal	Fehlercode Hexadezimal	Beschreibung
129	16#81	Unbekannte Verbindungs-Id
130	16#82	Unzulässige Länge
131	16#83	Nur zyklische Daten sind auf dieser Verbindung erlaubt.
132	16#84	Verbindung ist momentan nicht verfügbar
133	16#85	Der Timeout-Wert ist zu groß
134	16#86	Interner Programmfehler
135	16#87	Konfigurationsfehler
136	16#88	Übertragene Daten passen nicht zur konfigurierten Datenstruktur.
137	16#89	Funktionsbaustein gestoppt
138	16#8A	Timeout aufgetreten oder Senden blockiert
139	16#8B	Ein Funktionsbaustein dieser Art ist bereits auf dieser Verbindung aktiv

Tabelle 252: Zusätzliche Fehlercodes

8.8.6 Verbindungszustand

Fehlercode Dezimal	Fehlercode Hexadezimal	Beschreibung
0	16#00	Verbindung OK
1	16#01	Verbindung geschlossen
2	16#02	Server wartet auf Verbindungsaufnahme
4	16#04	Client versucht Verbindungsaufbau
8	16#08	Verbindung ist blockiert

Tabelle 253: Verbindungszustand

8.8.7 Partner Verbindungszustand

Protokollzustand Dezimal	Beschreibung
0	Keine Verbindung
1	Verbindung OK

Tabelle 254: Partner Verbindungszustand

9 SNTP-Protocol

(Simple Network Time Protocol)

Mit dem SNTP-Protokoll wird über Ethernet die Uhrzeit der SNTP-Clients durch den SNTP-Server synchronisiert.

HIMax Steuerungen können als **SNTP-Server** und/oder als **SNTP-Client** konfiguriert und eingesetzt werden. Es gilt der SNTP Standard nach RFC 2030 (SNTP-Version 4) mit der Einschränkung, dass nur der Unicast-Modus unterstützt wird.

Benötigte Ausstattung und Systemanforderung:

Element	Beschreibung
Steuerung	HIMax mit COM-Modul oder nur CPU-Modul
Aktivierung	Diese Funktion ist bei allen HIMax Systemen standardmäßig freigeschaltet.
Schnittstelle	Ethernet 10/100/1000BaseT

Tabelle 255: Systemanforderung und Ausstattung S&R TCP

9.1 SNTP-Client

Der SNTP-Client benutzt zu seiner Zeitsynchronisation immer nur den erreichbaren SNTP-Server mit der höchsten Priorität.

In jeder Ressource kann ein SNTP-Client zur Zeitsynchronisation konfiguriert werden.

So legen Sie einen neuen SNTP-Client an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Rechtsklick auf **Protokolle** und im Kontextmenü **Neu**, **SNTP-Client** wählen.
   ☑ Eine neuer SNTP-Client wird hinzuzugefügt.
3. Im Kontextmenü vom SNTP-Client **Eigenschaften** das **COM-Modul** auswählen.

Das Dialogfenster des SNTP-Client enthält die folgenden Parameter.

Element	Beschreibung
Typ	SNTP-Client
Name	Name für den SNTP-Client maximal 32 Zeichen.
Modul	Auswahl des CPU- oder COM-Moduls, auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert: Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen. Deaktiviert: Kein Limit der CPU-Last für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last des Moduls, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
Beschreibung	Beliebige eindeutige Beschreibung für den SNTP

Aktuelle SNTP-Version	Anzeige der aktuellen SNTP Version.
Referenz Stratum	Das Stratum eines SNTP-Clients gibt die Genauigkeit seiner lokalen Zeit wieder. Je niedriger das Stratum, desto genauer ist seine lokale Zeit. Null bedeutet ein unspezifiziertes oder nicht verfügbares Stratum (nicht gültig). Der aktuell verwendete SNTP-Server eines SNTP-Clients ist der, welcher erreichbar ist und die höchste Priorität besitzt. Ist das Stratum des aktuellen SNTP-Servers kleiner als das des SNTP-Clients, so übernimmt die Ressource die Zeit des aktuellen SNTP-Servers.
Referenz Stratum (Fortsetzung)	Ist das Stratum des aktuellen SNTP-Servers größer als das des SNTP-Clients, so übernimmt die Ressource die Zeit des aktuellen SNTP-Servers nicht. Ist das Stratum des aktuellen SNTP-Servers gleich dem des SNTP-Clients, so sind zwei Fälle zu unterscheiden: ▪ Wenn der SNTP-Client (Ressource) ausschließlich als SNTP-Client arbeitet, so übernimmt die Ressource die Zeit des aktuellen SNTP-Servers. ▪ Wenn der SNTP-Client (Ressource) gleichzeitig auch als SNTP-Server arbeitet, wird pro Anfrage des SNTP-Clients die Hälfte der Zeitdifferenz zum aktuellen-SNTP-Server auf der Ressource übernommen (Zeit nähert sich langsam an). Wertebereich: 1 ... 16 Standardwert: 15
Client Zeitanfrage Intervall [s]	Zeitintervall, in dem die Zeitsynchronisation durch den aktuellen SNTP-Server erfolgt. Das Client Zeitanfrage Intervall im SNTP-Client muss größer sein, als das Timeout im SNTP-Server. Wertebereich: 16s ... 16384s Standardwert: 16

Tabelle 256: Eigenschaften des SNTP-Client

9.2 SNTP-Client (Server Info)

In der SNTP-Server Info wird die Verbindung zu einem SNTP-Server konfiguriert.

Unterhalb eines SNTP-Clients könnnen 1 bis 4 SNTP-Server Infos konfiguriert werden.

So legen Sie einen neuen SNTP- Server Info an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle, SNTP Client** öffnen.
2. Rechtsklick auf **Protokolle** und im Kontextmenü **Neu**, **SNTP-Server Info** wählen.
   ☑ Eine neue SNTP-Server Info wird hinzuzugefügt.
3. Im Kontextmenü vom SNTP-Server Info **Eigenschaften** das **COM-Modul** auswählen.

Das Dialogfenster der SNTP-Server Info enthält die folgenden Parameter.

Element	Beschreibung
Typ	SNTP-Server-Info
Name	Name für die SNTP-Server-Info. Maximal 32 Zeichen.
Beschreibung	Beschreibung für den SNTP-Server. Maximal 31 Zeichen.
IP Adresse	IP-Adresse der Ressource oder des PC's, auf dem der SNTP-Server konfiguriert ist. Standardwert: 0.0.0.0
SNTP-Server Priorität	Priorität mit welcher der SNTP-Client diesen SNTP-Server behandelt. Die für einen SNTP-Client konfigurierten SNTP-Server sollten unterschiedliche Prioritäten besitzen. Wertebereich: 0 (geringste Priorität) bis 4294967295 (höchste Priorität.) Standardwert: 1
SNTP-Server Timeout [s]	Das Timeout im SNTP-Server muss kleiner eingestellt sein als das *Zeitanfragen Intervall* im SNTP-Client. Wertebereich: 1s ... 16384s Standardwert: 1

Tabelle 257: Eigenschaften SNTP-Server-Info

9.3 SNTP-Server

Der SNTP-Server nimmt die Anforderung von einem SNTP-Client entgegen und sendet seine aktuelle Zeit an den SNTP-Client zurück.

i Zeitsynchronisation einer Remote I/O durch eine HIMax Steuerung.
Auf dem HIMax Kommunikationsmodul, mit welchem die Remote I/O verbunden ist, muss ein SNTP Server eingerichtet sein.

So legen Sie einen neuen SNTP-Server an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Rechtsklick auf **Protokolle** und im Kontextmenü **Neu**, **SNTP-Server** wählen.
   ☑ Eine neuer SNTP Server wird hinzuzugefügt.
3. Im Kontextmenü vom SNTP Server **Eigenschaften** das **COM-Modul** auswählen.

Das Dialogfenster des SNTP Servers enthält die folgenden Parameter.

Element	Beschreibung
Typ	SNTP-Server
Name	Name für den SNTP-Server maximal 31 Zeichen.
Modul	Auswahl des CPU- oder COM-Moduls, auf dem dieses Protokoll abgearbeitet wird.
Max. CPU Load verwenden	Aktiviert : Limit der CPU-Last aus dem Feld Max. CPU Load [%] übernehmen. Deaktiviert: Kein Limit der CPU-Last, für dieses Protokoll verwenden.
Max. CPU Load [%]	Maximale CPU-Last das Modul, welche bei der Abarbeitung des Protokolls produziert werden darf. Wertebereich: 1 ... 100% Standardwert: 30%
Beschreibung	Beschreibung für den SNTP
Aktuelle SNTP-Version	Anzeige der aktuellen SNTP
Stratum des Zeitservers	Das Stratum eines SNTP-Servers gibt die Genauigkeit seiner lokalen Zeit wieder. Je niedriger das Stratum, desto genauer die lokale Zeit. Null bedeutet ein unspezifiziertes oder nicht verfügbares Stratum (nicht gültig). Das Stratum des SNTP-Servers muss niedriger oder gleich dem Stratum des anfragenden SNTP-Clients sein. Ansonsten wird die Zeit des SNTP-Servers vom SNTP-Client nicht übernommen. Wertebereich: 1 ... 15 Standardwert: 14

Tabelle 258: Eigenschaften SNTP-Server

10 X-OPC-Server

Der HIMA X-OPC-Server dient als Übertragungsschnittstelle zwischen HIMax Steuerungen und Fremdsystemen, die über eine OPC Schnittstelle verfügen.

OPC steht für Openess, Productivity & Collaboration und basiert auf der von Microsoft entwickelten Technologie (COM/DCOM). Damit lassen sich Prozessleitsystem, Visualisierungssysteme und Steuerungen der unterschiedlichsten Anbieter miteinander verbinden, um Daten auszutauschen, siehe auch www.opcfoundation.org.
Der HIMA X-OPC-Server wird nach der Installation auf einen PC als Windows-Dienst ausgeführt.

i Die gesamte Konfiguration und die Bedienung des X-OPC-Servers führen Sie in SILworX durch. Im SILworX Control Panel können Sie den X-OPC-Server wie eine Steuerung laden, starten und stoppen.

Der X-OPC-Server unterstützt die folgenden Spezifikationen:

- **Data Access (DA) Versionen 1.0, 2.05a und 3.0**
  DA wird für die Übertragung von Prozessdaten von der HIMax Steuerung zum OPC-Client genutzt.
  Jede Globale Variable der HIMax Steuerung können Sie zu einem OPC-Client übertragen.
- **Alarm&Event (A&E) Version 1.10**
  A&E wird zur Übertragung von Alarmen und Ereignissen von der HIMax Steuerung zum OPC-Client genutzt. Jede Globale Variable der HIMax Steuerung können Sie mit der Ereignisaufzeichnung überwachen.
  Ereignisse sind Änderungen des Zustands einer Variablen durch die Anlage oder Steuerung, die mit einem Zeitstempel versehen ist.
  Alarme sind solche Ereignisse, die eine Erhöhung des Gefahrenpotentials signalisieren.
  Es wird zwischen boolschen und skalaren Ereignissen unterschieden, siehe Kapitel 10.7.

10.1 Benötigte Ausstattung und Systemanforderung

Element	Beschreibung
Aktivierung	Die Freischaltung erfolgt per Software-Freischaltcode, siehe Kapitel 3.5. Die folgende Lizenzen können einzeln aktiviert werden: ▪ Data Access (DA) Server ▪ Alarm and Events (A&E) Server
PC Betriebssystem	Der X-OPC-Server ist auf einem x86 basierten PC unter folgenden Betriebsystemen lauffähig: ▪ Windows XP Professional (mind. Service Pack 2) (32Bit) ▪ Windows Server 2003 (32Bit) ▪ Windows Vista Ultimate (32Bit) ▪ Windows Vista Business (32Bit)
Anforderungen an den Host-PC	Minimalanforderung an den Host-PC: ▪ Pentium 4 ▪ 1 GByte (XP) bzw. 2,5 GByte RAM (Vista) ▪ Netzwerkkarte muss entsprechend dem Datenaufkommen ausgelegt sein 100Mbit/s oder 1Gbit/s i Die Minimalanforderungen gelten nur für den Betrieb eines X-OPC-Servers, wenn keine weiteren Anwendungen (z. B. SILworX, Word usw.) auf dem Host-PC betrieben werden.

Tabelle 259: Systemanforderung und Ausstattung des X-OPC-Server

10.2 Eigenschaften des X-OPC-Servers

Element	Beschreibung
OPC-Server	Der X-OPC-Server unterstützt die Funktionen ▪ OPC Data Access Custom Interface in den Versionen 1.0, 2.05a sowie 3.0. ▪ OPC Alarm & Event Interfaces 1.10
Sicherheitsgerichtet	Der X-OPC-Server läuft auf einem PC und ist nicht Sicherheitsgerichtet.
Schnittstelle	Empfohlen: Ethernet 1GBit/s
Datenaustausch	Datenaustausch über safe**ethernet**.
Ethernet Netzwerk	Die Netzwerkgeschwindigkeit des zugrundeliegenden Ethernet Netzwerks muss dem Datenaufkommen entsprechend ausgelegt sein (min. 100 Mbit/s, empfohlen 1GBit/s).
Globale Variablen	Es können Globale Variablen aus dem Kontext der Konfiguration verwendet werden.
Erlaubte Variablentypen	Alle Datentypen, die in SILworX angelegt werden können sind erlaubt.
Unerlaubte ASCII Zeichen	Die folgenden Zeichen sind reserviert und dürfen nicht (z.B. für Globale Variablen) verwendet werden: ! " # ' , . / \` : |
HIMax Steuerungen	Maximal 255 HIMax Steuerungen können von einem X-OPC-Server unterstützt werden.
Safeethernet Verbindung	Der X-OPC-Server kann 128 kB je safe**ethernet** Verbindung austauschen.
X-OPC-Server	10 X-OPC-Server können auf einem Host-PC betrieben werden.
X-OPC-Clients	10 OPC-Clients kann ein X-OPC-Server unterstützen.
Data Access Tags	Ein Data Access Server unterstützt maximal 100 000 DA Tags. Definition: Tags: Vom X-OPC-Server bereitgestellte Daten. Tags entsprechen den definierten Globalen Variablen. Items: Vom OPC-Client angeforderte Daten.
Alarm & Event Ereignisdefinitionen	Ein X-OPC Alarm & Event Server unterstützt maximal 100 000 Ereignisdefinitionen.

Tabelle 260: Eigenschaften des X-OPC-Server

10.3 Eigenschaften der HIMax Steuerung

Element	Beschreibung
Safeethernet Verbindung	Die HIMax kann insgesammt 128 kB je safe**ethernet** Verbindung zu einem X-OPC Server austauschen. Pro HIMax-Zyklus wird jedoch nur ein View zu einem X-OPC Server gesendet. (Ein View ist ein Fragment von 1100 Byte).
Schnittstellen	Ethernet 10/100/1000BaseT CPU-Module und COM-Module Verwendete Ethernet-Schnittstellen simultan auch für andere Protokolle nutzbar.
Max. Anzahl Ereignisdefinitionen	Auf einer HIMax Steuerung können maximal 20 000 Systemereignisse und 6000 E/A Ereignisse definiert werden.
Größe des Ereignisspeichers	Der nichtflüchtigen Ereignis-Puffer des HIMax Prozessormoduls erfasst maximal 5000 Ereignisse. Ist der Ereignis-Puffer voll, werden keine neuen Ereignisse gespeichert, bis ein Ereigniseintrag von mindestens einem X-OPC A&E Server ausgelesen und damit zum Überschreiben markiert wurde.
Alarm & Event Zeitstempel	Die Event Quelle kann für jedes Ereignis ausgewählt werden. Die als CPU Event definierten Ereignisse werden auf dem Prozessormodul gebildet. Dieses führt die Ereignisbildung komplett in jedem seiner Zyklen durch. Damit kann der Wert von jeder Globalen Variable als Ereignis erfasst und ausgewertet werden. Die als I/O Event definierten Ereignisse können nur auf SOE E/A Modulen (z.B. AI 32 02 oder DI 32 04) gebildet werden. Dieses führt die Ereignisbildung komplett in jedem seiner Zyklen durch. Wertebereich des Zeitstempels UTC (Universal Time Coordinated): sec Anteil seit 1970 in [udword] ms Anteil der Sekunde als [udword] von 0-999 Standardwert: 01.01.2000 / 00:00:00 Uhr Eine automatische Sommer-, Winterzeitumstellung wird nicht unterstützt.
Max. Anzahl X-OPC A&E Server	Maximal 4 X-OPC A&E Server können auf die HIMax Steuerung zugreifen und Ereignisse simultan aus dem Ereignis-Puffer des Prozessormoduls lesen.

10.4 Erforderliche Aktionen bei Änderungen

Die folgende Tabelle zeigt die Aktionen, die nach einer Änderung in den einzelnen Systemen durchgeführt werden müssen.

Art der Änderung	Änderungen bei		
	HIMax	HIMatrix	X-OPC
DA			
Tags Zufügen	C+D	C+D	C+D
Tagnamen (GV Namensänderung)	C+D	C+D	C+D
Tags löschen	C+D	C+D	C+D
Views ändern (Parameter und Zufügen/Löschen)	C+D	C+D	C+D
A&E			
Event Definition zufügen	C+D	n. a.	C+D
Event Definition löschen	C+D	n. a.	C+D
Ändern Event Source	C+D	n. a.	C+D
Ändern Alarm Texts	-	n. a.	C+D
Ändern Alarm Severity	-	n. a.	C+D
Ändern Parameter "ACK Required"	-	n. a.	C+D
Ändern "Alarm Values" bei skalaren Event	C+D	n. a.	C+D
Ändern Parameter "Alarm at False" bei boolean Events	C+D	n. a.	C+D
Name ändern	C+D	n. a.	C+D
E/A Kanal mit GV verbinden	C+R	n. a.	-
Zustandsvariablen mit GV verbinden	C+R	n. a.	-
Generell			
Ändern safe**ethernet** Parameter	C+D	C+D	C+D

Tabelle 261: Erforderliche Aktionen bei Änderungen

C: Codegenerierung erforderlich

R: Reload erforderlich

D: Download erforderlich

n. a: nicht anwendbar

-: keine Aktion nötig

10.5 Forcen von Globalen Variablen auf E/A Modulen

i Werden von einem E/A Modul die globale Variablen, die mit dem Prozesswert verbunden sind geforcet, wirken diese nicht auf die Globalen Variablen die mit den Parameteren **->Zustand- LL, -L,- -N, -H, -HH** verbunden sind.
Dies gilt auch dann, wenn diese Alarme im Alarm&Event Editor eingetragen sind.
Beim Test müssen diese globalen Variablen einzeln geforcet werden.

10.6 Konfiguration einer OPC-Server Verbindung

In diesem Beispiel wird eine redundante X-OPC-Server Verbindung mit einer HIMax Steuerung konfiguriert.

Die X-OPC-Server stellen die Prozessvariablen und Ereigniswerte der HIMax Steuerung für die OPC-Clients bereit. Die OPC-Clients greifen auf die bereitgestellten Prozessvariablen und Ereigniswerte zu und stellen diese auf ihrer Benutzeroberfläche dar.

10.6.1 Benötigte Software:

- SILworX
- X-OPC-Server
- OPC-Client

i Die gesamte Konfiguration und die Bedienung des X-OPC-Servers führen Sie in SILworX durch. Im SILworX Control Panel können Sie den X-OPC-Server wie eine Steuerung laden, starten und stoppen.

10.6.2 Bedingungen für den X-OPC-Server Betrieb:

- Das Ethernet-Netzwerk sollte eine Bandbreite von mindestens 100 Mbit/s (besser 1GBit/s) haben.
- Die Systemzeit der Rechner/Server sollte synchronisiert werden, z. B. mittels SNTP.
- Stellen Sie sicher, daß die Datensätze für Data Access und Alarm & Events auf der Steuerung, den X-OPC-Servern und den OPC-Clients zueinander passen.

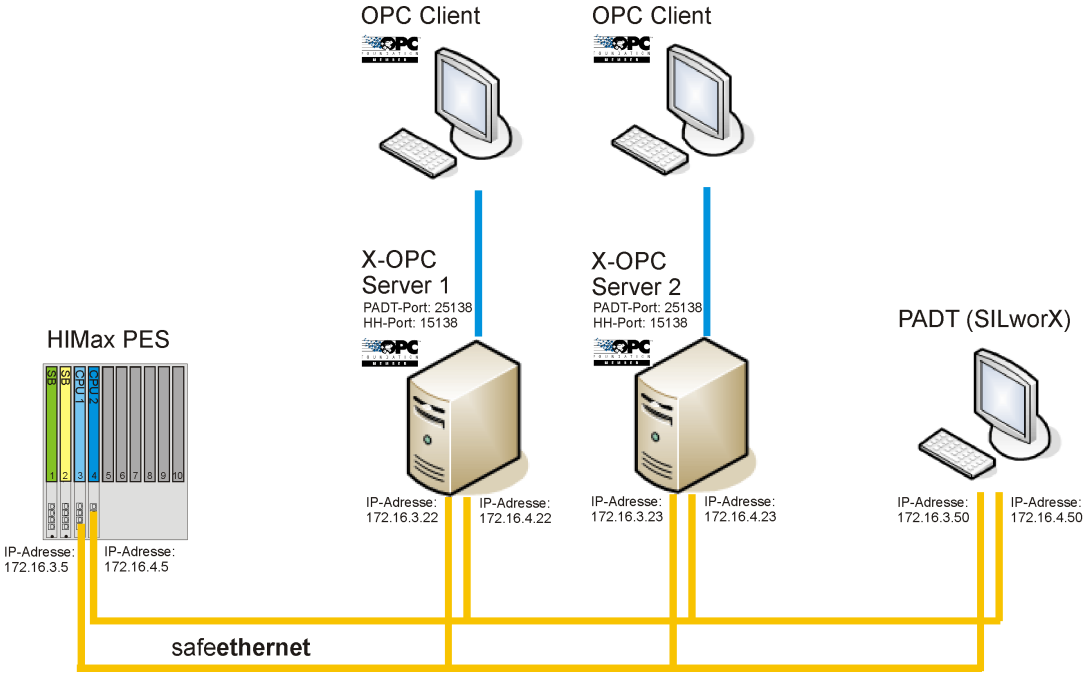


Bild 69: Redundanter X-OPC Betrieb

10.6.3 Installation auf dem Host-PC

Der X-OPC-Server muss auf dem jeweiligen Host-PC installiert werden.

I Notieren Sie sich die System ID und die Nummer des PADT Port. Diese werden zur Generierung des Lizenzschlüssels benötigt!

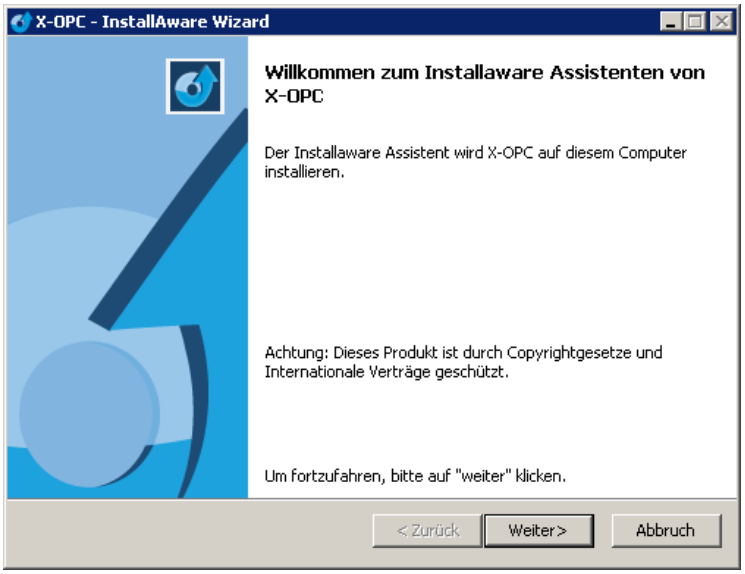

X-OPC - InstallAware Wizard
Willkommen zum Installaware Assistenten von X-OPC
Der Installaware Assistent wird X-OPC auf diesem Computer installieren.
Achtung: Dieses Produkt ist durch Copyrightgesetze und Internationale Verträge geschützt.
Um fortzufahren, bitte auf "weiter" klicken.
< Zurück
Weiter >
Abbruch

Bild 70: Installationsroutine des X-OPC-Servers

So installieren Sie den X-OPC-Server auf dem ersten Host-PC:

Starten Sie die Datei *X-OPC.exe* auf dem jeweiligen Host-PC und folgen Sie der Installationsroutine.

1. Folgende Daten für den X-OPC-Server eingeben:
   - System ID: 100
   - PADT Port: 25138
   - Frei wählbarer Name des X-OPC-Servers (wird im OPC-Client angezeigt).
2. Zum installieren des X-OPC-Servers **Weiter>** klicken.

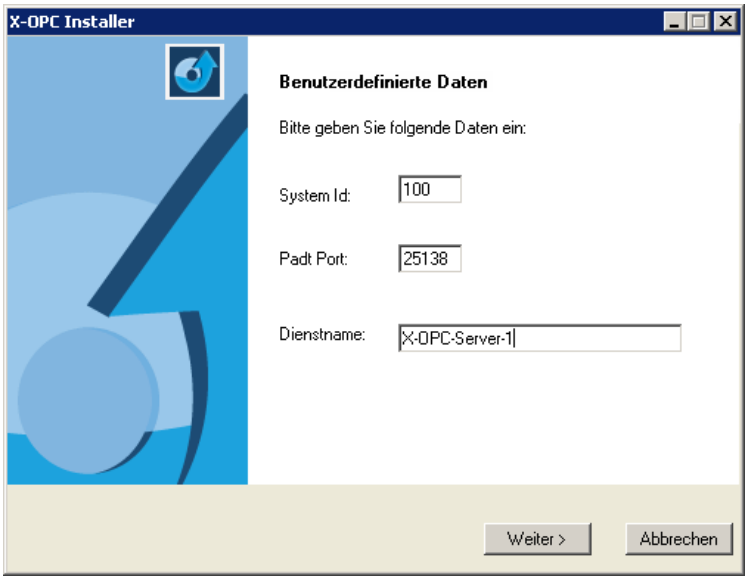


Bild 71: Installationsroutine des X-OPC-Servers

So installieren Sie den X-OPC-Server auf dem zweiten Host-PC:

i Ermitteln Sie zuerst die ClassID des ersten X-OPC-Server, bevor Sie den zweiten X-OPC Server installieren!

Bei der redundanten Verbindung eines OPC-Client mit zwei X-OPC Server, erwarten einige OPC-Client Systeme, dass die ClassID der beiden X-OPC Server gleich ist. Ermitteln Sie zunächst die ClassID des ersten X-OPC Servers (z.B. mit Hilfe des OPC-Client) und notieren Sie sich diese.

Starten Sie die Datei X-OPC.exe auf dem zweiten Host-PC und folgen Sie der Installationsroutine.

1. Folgende Daten für den X-OPC-Server eingeben:
   - System ID: 110
   - PADT Port: 25138
   - Frei wählbarer Name des X-OPC-Servers (wird im OPC-Client angezeigt).

i PADT Port und HH Port des zweiten X-OPC-Servers dürfen gleich dem ersten sein, wenn die X-OPC-Server auf unterschiedlichen PCs betrieben werden.

2. Zur Bestätigung **Weiter>** klicken.

So stellen Sie die gleiche ClassID auf dem zweiten Host-PC ein:

1. Wählen Sie CLSID Einstellung **manuell** für DA und AE.
2. Tragen Sie die ClassID des ersten X-OPC-Servers in die Felder **CLSID** ein.
3. Zum installieren des X-OPC-Servers **Weiter>** klicken.

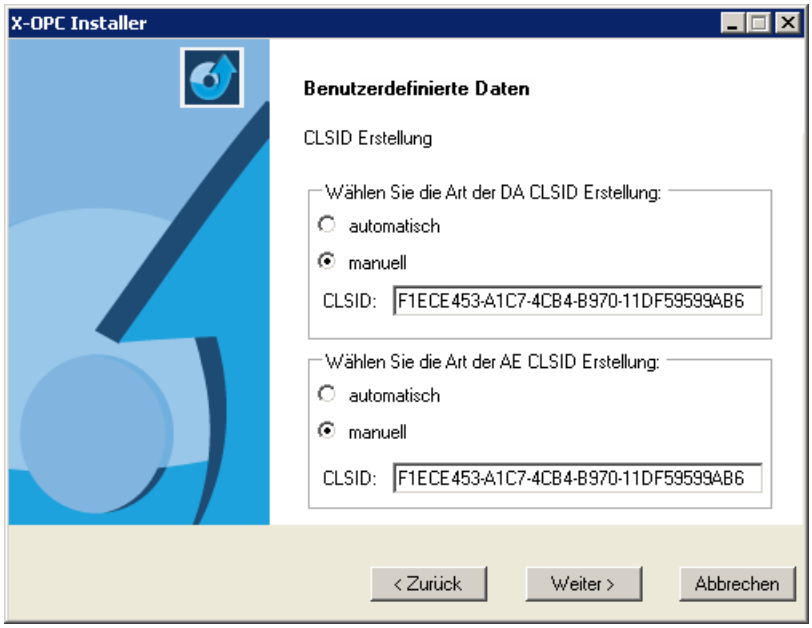


Bild 72: Mauelle Einstellung der ClassID des zweiten X-OPC-Servers

Automatisches starten der X-OPC-Server nach einem Neustart des PCs:

1. In Windows **Start, Einstellung, Systemsteuerung, Verwaltung, Dienste** öffnen und **X-OPC-Server** aus der Liste wählen.
2. Im Kontextmenü des X-OPC-Servers **Eigenschaften** wählen.
3. Im Register **Allgemein** den Starttyp **Automatisch** wählen.

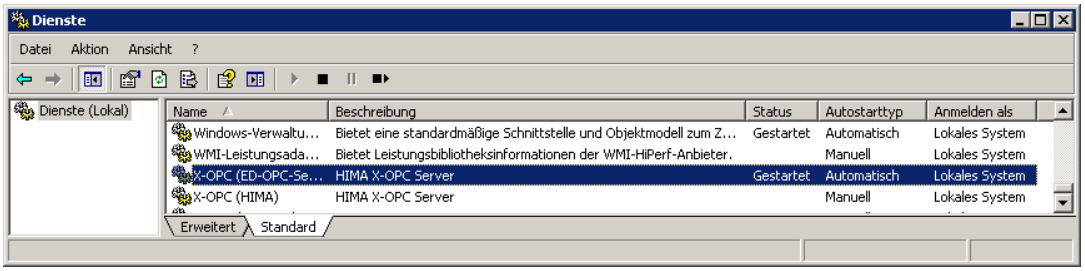


Bild 73: Einstellung für X-OPC-Server automatisch starten

So prüfen Sie, ob der X-OPC-Server auf dem PC läuft:

1. *Windows Task-Manager* öffnen und Register *Prozesse* wählen.
2. Prüfen, ob der Prozess *X-OPC.exe* auf dem PC läuft.

i Wenn OPC-Client und OPC-Server nicht auf dem gleichen PC laufen, dann müssen Sie die DCOM Schnittstelle anpassen.

Dazu sind die Schritte auszuführen, die im Handbuch der OPC Foundation *Using OPC via DCOM with Microsoft Windows XP Service Pack 2 Version 1.10* (siehe www.opcfoundation.org) beschrieben werden.

10.6.4 OPC Server in SILworX konfigurieren

So legen Sie in SILworX einen neuen OPC-Server-Set an:

1. Im Strukturbaum **Konfiguration** öffnen.
2. Rechtsklick auf **Konfiguration** und im Kontextmenü **Neu**, **OPC-Server-Set** wählen.
   ☑ Ein neuer OPC-Server-Set wird hinzugefügt.
3. Im Kontextmenü vom OPC-Server-Set **Eigenschaften** wählen und Standardwerte übernehmen.

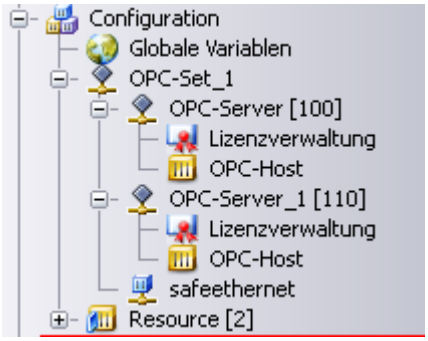


Bild 74: Redundanter X-OPC Betrieb

So konfigurieren Sie in SILworX den ersten OPC-Server:

1. Im Strukturbaum **Konfiguration**, **OPC-Server-Set** auswählen.
2. Rechtsklick auf **OPC-Server-Set** und im Kontextmenü **Neu**, **OPC-Server** wählen.
   ☑ Ein neuer OPC-Server wird hinzugefügt.
3. Rechtsklick auf **OPC-Server** und im Kontextmenü **Eigenschaften** wählen.
   - System ID [SRS] ein (z. B. 100) eintragen.
   - Standardeinstellungen übernehmen.
3. Rechtsklick auf **OPC-Host** und im Kontextmenü **Edit** wählen.
   ☑ OPC-Host Dialog zur Konfiguration der IP-Schnittstellen wird geöffnet
4. Rechtsklick auf eine leere Stelle im OPC-Host Dialog und im Kontextmenü **Neuer IP-Anschluss** wählen.
   - PADT-Port einstellen (z. B. 25138).
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.3.22).
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.4.22).
   - Als Standardschnittstelle markieren.
   - HH-Port einstellen (z. B. 15138).

Im gleichen OPC-Server-Set konfigurieren Sie den redundanten OPC Server an.

So konfigurieren Sie den zweiten OPC-Server:

1. Im Strukturbaum **Konfiguration**, **OPC-Server-Set** auswählen.
2. Rechtsklick auf **OPC-Server-Set** und im Kontextmenü **Neu**, **OPC-Server** wählen.
   ☑ Ein neuer OPC-Server wird hinzugefügt.
3. Rechtsklick auf **OPC-Server** und im Kontextmenü **Eigenschaften** wählen.
   - System ID [SRS] ein (z. B. 110) eintragen.
   - Standardeinstellungen übernehmen
5. Rechtsklick auf **OPC-Host** und im Kontextmenü **Edit** wählen.
   ☑ OPC-Host Dialog zur Konfiguration der IP-Schnittstellen wird geöffnet
6. Rechtsklick auf eine leere Stelle im OPC-Host Dialog und im Kontextmenü **Neuer IP-Anschluss** wählen.
   - PADT-Port einstellen (z. B. 25138).
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.3.23).
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.4.23).
   - Als Standardschnittstelle markieren.
   - HH-Port einstellen (z. B. 15138).

i Ist auf dem PC eine Firewall installiert, dann müssen in der Konfiguration der Firewall die TCP/UDP PADT- und HH-Ports des X-OPC-Servers als Ausnahmen eingetragen werden.

10.6.5 Einstellungen des OPC-Server im safeethernet Editor

So erstellen Sie die safeethernet Verbindung zwischen OPC-Server und Ressource (HIMax Steuerung):

1. Im OPC-Server-Set den **safeethernet Editor** öffnen
2. In der Objektauswahl auf die **Ressource** klicken und per Drag&Drop auf eine freie Stelle im Arbeitsbereich des safe**ethernet** Editors ziehen.
3. Für Alarm & Events den Parameter *SER aktivieren* standardmäßig aktiviert.

safeethernet OPC Server-Set_1 *

	Partner	IF CH 1 (lokal)	IF CH 2 (lokal)	IF CH 1 (Ziel)	IF CH 2 (Ziel)	Profil	Sync/Async	R
1	Ressource					Fast & Noisy	ASYNC	
2	Ressource	100.x.x (172.16.3.22:15138)	100.x.x (172.16.3.22:15138)	2.0.5 (172.16.3.5:6010)	2.0.5 (172.16.3.5:6010)			
3	Ressource	110.x.x (122.16.4.23:15138)	110.x.x (122.16.4.23:15138)	2.0.15 (172.16.4.5:6010)	2.0.15 (172.16.4.5:6010)			

Bild 75: Redundanter X-OPC Betrieb

i Die verwendete Ethernet Schnittstellen der PCs werden in der Spalte **IF CH1 (lokal)** abgebildet. Die Ethernet Schnittstellen der HIMax Steuerung müssen in der Spalte **IF CH1 (Ziel)** ausgewählt werden.

Die Standardwerte der safe**ethernet** Parameter für die X-OPC-Server Kommunikation sind auf die maximale Verfügbarkeit ausgelegt.

Receive Timeout = 1000 ms, Response Time = 500 ms usw.

Informationen zu den safe**ethernet** Parameter, siehe Kapitel 4.6.

10.6.6 X-OPC Data Access Server in SILworX konfigurieren

So erstellen Sie die View-Definitionen der safeethernet Verbindung:

Voraussetzung: Der safe**ethernet**-Editor des OPC-Servers muss geöffnet sein.

1. Rechtsklick auf die Zeile der **Ressource** um das Kontextmenü der Ressource zu öffnen.
2. Im Kontextmenü **Detailansicht** wählen, um die Detailansicht der safe**ethernet**-Verbindung zu öffnen.
3. Klick auf Register **View Definitionen**.
4. Rechtsklick auf eine frei Stelle im Arbeitsbereich und **Neue View-Definition** wählen. In der Spalte Priorität wird eingestellt, wie oft dieser View im Verhältnis zu den anderen Views gesendet wird (Ein View ist ein Fragment der Größe 1100 Byte). Verwenden Sie für die View-Definitionen zunächst die Standardeinstellung mit der Priorität 1, siehe auch Kapitel 10.6.8.
5. Klick auf Register **OPC-Server Set<->Ressource**.

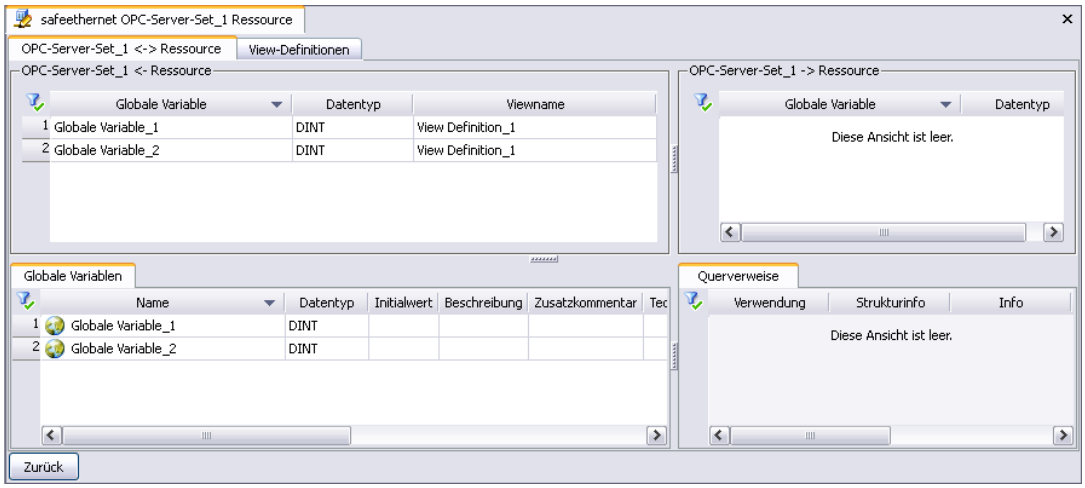


Bild 76: Detailansicht der safe**ethernet**-Verbindung

So werden die OPC-Empfangsvariablen hinzugefügt:

OPC-Empfangsvariablen werden von der Ressource zum OPC-Server gesendet.

1. Detailansicht des X-OPC safe**ethernet**-Editors öffnen und Register **OPC-Server Set<->Ressource** wählen.
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in den Bereich **OPC-Server Set <-Ressource-** ziehen.
3. Doppelklick auf Spalte **Viewname** und zuvor angelegte **View Definition** wählen.
4. Diesen Schritt für weitere OPC-Empfangsvariablen wiederholen.

So werden die OPC-Sendevariablen hinzugefügt:

OPC-Sendevariablen werden vom OPC-Server zur Ressource gesendet

1. Detailansicht des X-OPC safe**ethernet**-Editors öffnen und Register **OPC-Server Set<->Ressource** wählen.
2. In der Objektauswahl eine **Globale Variable** wählen und per Drag&Drop in den Bereich **OPC-Server Set->Ressource** ziehen.
3. Doppelklick auf Spalte **Viewname** und zuvor angelegte **View Definition** wählen.
4. Diesen Schritt für weitere OPC-Sendevariablen wiederholen.

i Die OPC Sende- und Empfangsvariablen müssen Sie im OPC-Server-Set nur einmal anlegen. Diese werden automatisch von beiden X-OPC-Server im OPC-Server-Set verwendet.

Codegenerierung und Laden der Ressource:

1. Im Strukturbaum **Konfiguration, Ressource** wählen.
2. Klick auf **Codegenerierung** in der **Action Bar** und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.
4. Laden Sie den generierten Code in die Ressource.

Codegenerierung und Verifikation des OPC-Server-Set:

1. Im Strukturbaum **Konfiguration, OPC-Server-Set** wählen.
2. Klick auf **Codegenerierung** in der **Action Bar** und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

Laden Sie den generierten Code in den X-OPC-Server:

1. Rechtsklick auf den **OPC-Server** und aus dem Kontextmenü **Online** zum **System-Login** wählen.
2. Zugangsdaten eingeben:
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.3.23).
   - Benutzername: Administrator
   - Passwort: ohne
   - Rechte: Administrator
3. Klick auf **Login** zum öffnen des Control-Panels.
4. In der SILworX Menüleiste auf das Symbol **Ressource Download** klicken.
   ☑ Code wird in den X-OPC-Server geladen.
5. Klicken Sie in der SILworX Menüleiste auf das Symbol **Ressource Kaltstart**.
   ☑ X-OPC-Server läuft.

Öffnen Sie den OPC-Client:

Der im OPC-Client angezeigte Name des X-OPC-Servers setzt sich zusammen aus: **HIMA** (Hersteller)**.Dienstname** (siehe Kapitel 10.6.3)**-DA** (Data Access).

Stellen Sie die Verbindung zum X-OPC-Server her. Die konfigurierten Data Access Daten sollten jetzt zum OPC-Client übertragen werden.

10.6.7 X-OPC Alarm & Event Server in SILworX konfigurieren

In diesem Beispiel wird ein X-OPC A&E Server zu einer HIMax Steuerung konfiguriert. Der X-OPC A&E Server erfasst die Ereignisse über safe**ethernet** von der HIMax Steuerung und stellt diese für den OPC-Client bereit. Der OPC-Client greift auf die bereitgestellten Ereignisvariablen zu und stellt diese auf seiner Benutzeroberfläche dar.

So legen Sie einen Alarm&Event Editor an:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf **Ressource** und im Kontextmenü **Neu, Alarm & Events** wählen.
   ☑ Der Alarm&Event Editor wird neu hinzugefügt.

So legen Sie die Alarm&Events an:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf **Alarm & Events** und **Edit** wählen.
3. Register **Event Definition Bool** für boolsche Ereignisse wählen, siehe Kapitel 10.7.1.
4. Register **Event Definition Skalar** für skalare Ereignisse wählen, siehe Kapitel 10.7.2.
5. In der Objektauswahl auf die **Globale Variable** klicken und per Drag&Drop auf eine freie Stelle im Arbeitsbereich des Alarm & Event Editors ziehen.
6. Die Priorität der Ereignisse geben Sie im safe**ethernet** Editor ein, siehe Kapitel 10.6.8.

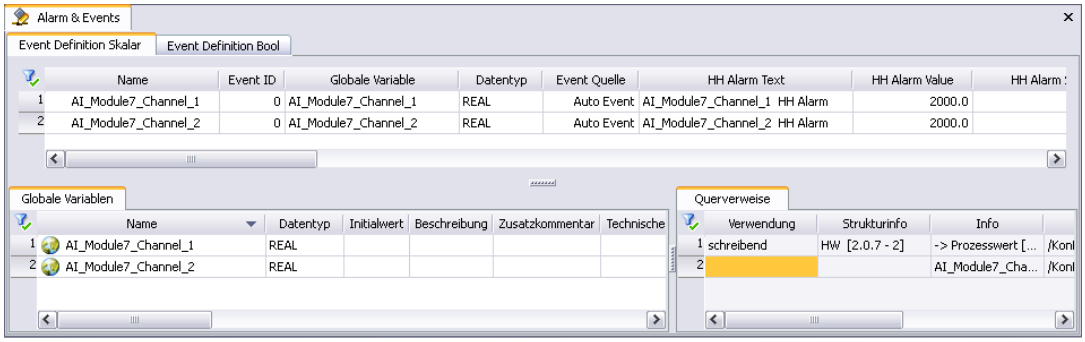


Bild 77: Alarm & Event Editor

So erstellen Sie die Acknowledge Verbindung zwischen den beiden Alarm&Event X-OPC-Servern:

i Werden zwei Alarm&Event X-OPC-Server redundant betrieben, können die Acknowledgements zur Bestätigung (Quittierung) der Alarme auf beiden X-OPC-Servern synchronisiert werden. Dazu wird eine Acknowledge Verbindung angelegt.

1. Im Strukturbaum **Konfiguration, OPC-Server-Set, Neu** wählen.
2. Rechtsklick auf **OPC-Server-Set** und im Kontextmenü **Neu**, **OPC-A&E-Ack** wählen.
3. Im OPC-A&E-Ack Dialog die folgenden IP-Verbindungen wählen.
   - IF CH1 (OPC-Server 1, z. B. 172.16.3.22).
   - IF CH2 (OPC-Server 1, z. B. 172.16.4.22).
   - IF CH1 (OPC-Server 2, z. B. 172.16.3.23).
   - IF CH2 (OPC-Server 2, z. B. 172.16.4.23).

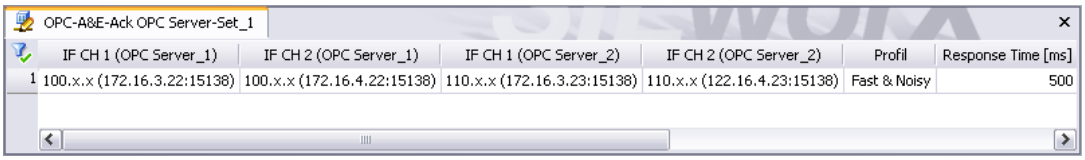


Bild 78: Redundanter X-OPC Betrieb

Codegenerierung und Laden der Ressource:

1. Im Strukturbaum **Konfiguration, Ressource** wählen.
2. Klick auf **Codegenerierung** in der **Action Bar** und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.
4. Laden Sie den generierten Code in die Ressource.

Codegenerierung und Verifikation des OPC-Server-Set:

1. Im Strukturbaum **Konfiguration, OPC-Server-Set** wählen.
2. Klick auf **Codegenerierung** in der **Action Bar** und mit **OK** bestätigen.
3. Einträge in der Statusanzeige sorgfältig überprüfen, gegebenenfalls korrigieren.

Laden Sie den generierten Code in den X-OPC-Server:

1. Rechtsklick auf den **OPC-Server** und aus dem Kontextmenü **Online** zum System-Login wählen.
2. Zugangsdaten eingeben:
   - IP-Adresse des PC, auf dem der X-OPC-Server installiert ist (z. B. 172.16.3.23).
   - Benutzername: Administrator
   - Passwort: ohne
   - Rechte: Administrator
3. Klick auf **Login** zum öffnen des Control-Panels.
4. In der SILworX Menüleiste auf das Symbol **Ressource Download** klicken.
   ☑ Code wird in den OPC-Server geladen.
5. Klicken Sie in der SILworX Menüleiste auf das Symbol **Ressource Kaltstart**.
   ☑ OPC-Server läuft.

Öffnen Sie den OPC-Client:

Der im OPC-Client angezeigte Name des X-OPC-Servers setzt sich zusammen aus: **HIMA** (Hersteller)**.Dienstname** (siehe Kapitel 10.6.3)**-AE (Alarm&Event)**.

Stellen Sie die Verbindung zum X-OPC-Server her. Die konfigurierten Alarme & Events sollten jetzt zum OPC-Client übertragen werden.

i Werden eine Steuerung und ein X-OPC A&E Server verbunden, muss sich der X-OPC A&E Server zunächst aufsynchronisieren. Dazu liest der X-OPC A&E Server von allen Variablen, die als Ereignis definiert sind, den aktuellen Zustand aus und überträgt die anstehenden Alarme zum OPC-Client. Im OPC-Client kann damit ein Abbild über den Zustand der Steuerung gebildet werden. Erst ab diesem Zeitpunkt werden die Ereignisse ausgelesen.

Nach dem Aufsynchronisieren des X-OPC-Servers auf die Steuerung werden auf dem OPC-Client alle Ereignisse aktualisiert. Ereigniseinträge mit einem älteren Zeitstempel werden mit den aktuell ausgelesenen Zuständen der Ereignisvariablen überschrieben.

10.6.8 Parametrierung der Views und Prioritäten in SILworX

Die HIMax kann insgesamt 128 kB je safe**ethernet** Verbindung zu einem X-OPC Server senden, jedoch nur 1100 Byte pro HIMax-CPU-Zyklus. Um mehr Daten über eine safe**ethernet** Verbindung zu senden werden die Daten fragmentiert. Mit dem Parameter Priorität dieser Fragmente (Views) können Sie bestimmen wie häufig diese Views aktualisiert werden sollen.

i Views mit der Priorität **n** und Views mit der Priorität **m** werden im Verhältnis **n** zu **m** mal versendet.

Für die Reaktionszeit von der Steuerung zum X-OPC Server ist zusätzlich die Anzahl der Views und Kommandos (z.B. Stopp, Start) von SOE zu berücksichtigen.

$T_R = t_1 + t_2 + t_3 + t_4$; gilt nur, wenn das Priorität aller Views für Zustandsdaten 1 ist

- T_R Worst Case Reaction Time
- t_1 Sicherheitszeit des PES 1
- t_2 *Anzahl Views * ReceiveTMO*
- t_3 Sicherheitszeit des X-OPC Servers
- t_4 Zeitverzug durch SOE Funktion; abhängig von Ereignisaufkommen und Verbindungsaufnahme

Für die umgekehrte Richtung kann die Reaktionszeit mit der selben Formel ermittelt werden, nur dass hierbei in der Regel nur eine View zum Tragen kommt, da der X-OPC Server nur die von OPC-Clients geschriebenen Daten transferiert.

Maximale Anzahl der Views: 1024

Maximale größe eines Views: 1100 Byte

Wertebereich der Prioritäten: 1 (höchste) bis 65 535 (niedrigste)

Bezeichnung	Standardwert der Priorität
Priorität Ereignisse (Alarm&Event)	1
Priorität Zustandswerte (Alarm&Event)	10
Priorität der View Definitionen (Data Access)	1

Tabelle 262: Standardwerte der Prioritäten

Prioritäten der Views für Ereignisse (Alarm&Event)

Die von Ihnen im Alarm&Event Editor angelegten Ereignisse werden automatisch fragmentiert und in Views übertragen.

Die Priorität für die Ereignisse geben Sie im safe**ethernet** Editor in den Spalten **Priorität Ereignisse** und **Priorität Zustandswerte** ein, diese Prioritäten gelten dann für alle Alarm&Event Views dieser safe**ethernet** Verbindung.

So stellen Sie die Prioritäten für die Alarm&Event Views ein

1. Im OPC-Server-Set den safe**ethernet** Editor öffnen

Bild 79: safe**ethernet** Editor

2. Doppelklick auf **Priorität Ereignisse**, um die Priorität der Ereignisse zu ändern. Die Views der Ereignisse erhalten alle die in der Spalte **Priorität Ereignisse** eingetragene Priorität (z.B. 1). Damit legt man fest, mit welcher Priorität der X-OPC Server Ereignisse von der Steuerung anfordert. Sind zu diesem Zeitpunkt in der Steuerung keine Ereignisse vorhanden werden auch keine übertragen.
3. Doppelklick auf **Priorität Zustandswerte**, um die Priorität der Zustandswerte der Ereignisse zu ändern. Die Views der Zustandswerte der Ereignisse erhalten alle die in der Spalte **Priorität Zustandswerte** eingetragene Priorität (z.B. 10).

i Die Zustandswerte der Ereignisse werden nur zur Synchronisation (z. B. bei Verbindungsaufbau) benötigt, und können daher in einem größeren Zeitabstand als die Ereignisse übertragen werden.

Prioritäten der Data Access Views

Für jedes Data Access View können Sie Globale Variablen zuweisen und die Priorität des Views einstellen, mit der Sie bestimmen wie oft diese Variablen aktualisiert werden.

So stellen Sie die Prioritäten für die Data Access Views ein

Voraussetzung: Der safe**ethernet**-Editor des OPC-Servers muss geöffnet sein.

1. Rechtsklick auf die Zeile der **Ressource** um das Kontextmenü der Ressource zu öffnen.
2. Im Kontextmenü **Detailansicht** wählen, um die Detailansicht der safe**ethernet**-Verbindung zu öffnen.
3. Klick auf Register **View Definitionen**.
4. In der Spalte Priorität stellen Sie die Priorität des Data Access Views ein.
   - Vergeben Sie für die Data Access Views mit Globalen Variablen die häufig aktualisiert werden sollen eine hohe Priorität (z.B. 1).
   - Vergeben Sie für die Data Access Views mit den Globalen Variablen die seltener aktualisiert werden sollen eine niedrigere Priorität (z.B. 10).

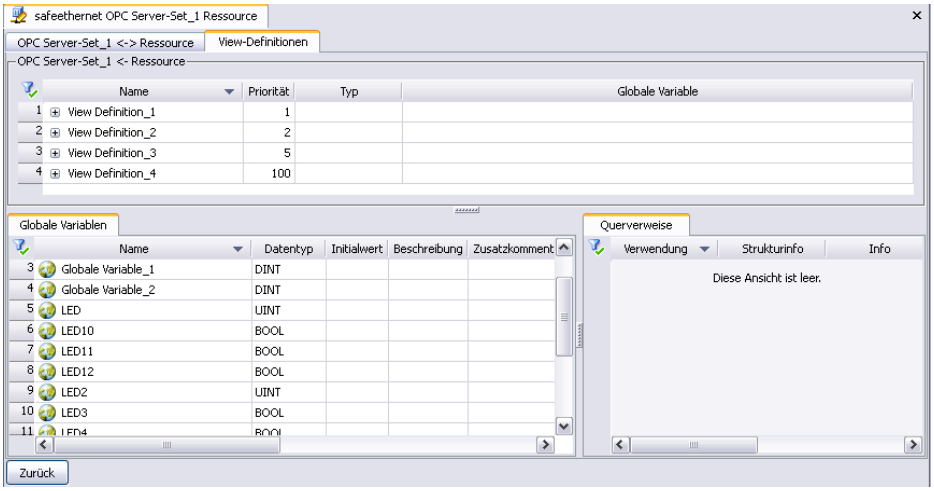


Bild 80: Detailansicht der safe**ethernet**-Verbindung

Zusätzlich kann der Zustand und der Zeitstempel von jedem Data Access View über die folgenden Variablen ausgelesen werden.

Name	Beschreibung
View-Zeitstempel [ms]	Millisekunden Anteil des Zeitstempels (aktuelle Systemzeit).
View-Zeitstempel [s]	Sekunden Anteil des Zeitstempels (aktuelle Systemzeit).
View-Zustand	<table><tr><th>Status</th><th>Beschreibung</th></tr><tr><td>0</td><td>CLOSED: Verbindung ist geschlossen.</td></tr><tr><td>1</td><td>TRY OPEN: Verbindung wird versucht zu öffnen, sie ist jedoch noch nicht geöffnet.</td></tr><tr><td>2</td><td>CONNECTED: Die Verbindung steht und aktuelle View-Daten wurden empfangen (vgl. Zeitstempel). Solange keine View-Daten empfangen werden, bleibt der View-Zustand beim Verbindungsaufbau auf TRY_OPEN.</td></tr></table> i Der Verbindungszustand des safe**ethernet** Editors (siehe Kapitel 4.4) wird auf CONNECTED gesetzt sobald die Verbindung geöffnet ist. Im Gegensatz zum View-Zustand müssen hier noch keine Daten ausgetauscht worden sein.

Tabelle 263: Zustand und der Zeitstempel der Data Access Views

10.7 Alarm & Event Editor

Der Alarm & Event Editor dient der Parametrierung der Alarm und Events der HIMax-Steuerung.

So legen Sie einen Alarm&Event Editor an:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf **Ressource** und im Kontextmenü **Neu, Alarm & Events** wählen.
   ☑ Ein neues Objekt Alarm & Events wird hinzuzugefügt.

So legen Sie die Alarm&Events an:

1. Im Strukturbaum **Konfiguration, Ressource** öffnen.
2. Rechtsklick auf **Alarm & Events** und **Edit** wählen.
3 Register **Event Definition Bool** für boolsche Ereignisse wählen, siehe Kapitel 10.7.1.
4. Register **Event Definition Skalar** für skalare Ereignisse wählen, siehe Kapitel 10.7.2.
5. In der Objektauswahl auf die **Globale Variable** klicken und per Drag&Drop auf eine freie Stelle im Arbeitsbereich des Alarm & Event Editors ziehen.
6. Die Priorität der Ereignisse geben Sie im safe**ethernet** Editor ein, siehe Kapitel 10.6.8.

HIMax unterscheidet boolsche und skalare Ereignisse.

10.7.1 Boolesche Ereignisse

- Änderungen von booleschen Variablen, z. B. von digitalen Eingängen.
- Alarm- und Normalzustand, diese können Sie den Zuständen der Variablen beliebig zuordnen

Die Parameter der **booleschen** Ereignisse geben Sie im Alarm & Event Editor der Ressource ein, der die folgenden Spalten enthält:

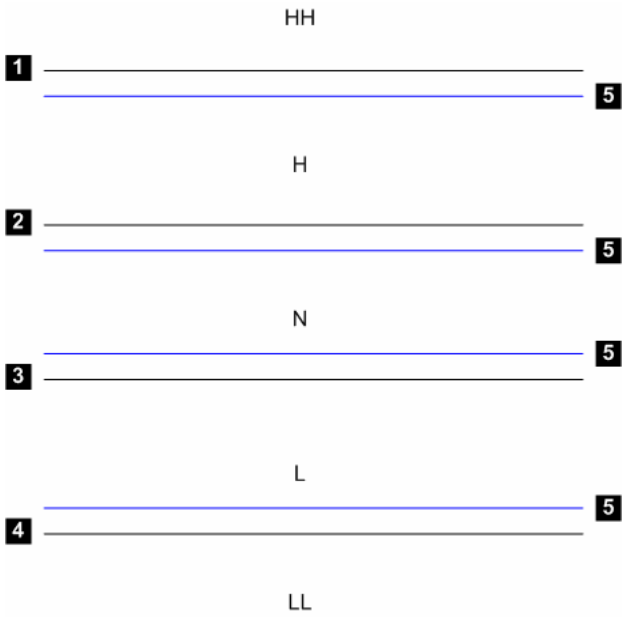
Spalte	Beschreibung	Wertebereich
Name	Name der Ereignisdefinition	Text, max. 31 Zeichen
Globale Variable	Name der zugewiesenen globalen Variable (Eingefügt z. B. durch Drag&Drop)	
Datentyp	Datentyp der globalen Variable, nicht änderbar	BOOL
Eventquelle	CPU Event: Der Zeitstempel wird auf einem Prozessormodul gebildet. Dieses führt die Ereignisbildung komplett in jedem seiner Zyklen durch. IO Event: Der Zeitstempel wird auf einem geeigneten E/A-Modul gebildet (z. B. DI 32 04). Auto Event: Es wird ein CPU Event und wenn vorhanden IO Events der E/A Module gebildet. Standardwert: CPU Event	CPU Event, IO Event, Auto Event
Alarm bei FALSE	Aktiviert: Die Wertänderung TRUE->FALSE der globalen Variablen löst ein Ereignis aus Deaktiviert: Die Wertänderung FALSE->TRUE der globalen Variablen löst ein Ereignis aus Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
Alarm Text	Text, der den Alarmzustand benennt	Text
Alarm Severity	Priorität des Alarmzustands Standardwert: 1	1...1000

Alarm Ack Required	Aktiviert Bestätigung des Alarmzustandes durch den Bediener erforderlich (Quittierung) Deaktiviert Bestätigung des Alarmzustandes durch den Bediener nicht erforderlich Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
Return to Normal Text	Text, der den Alarmzustand benennt	Text
Return to Normal Severity	Priorität des Normalzustands	1...1000
Return to Normal Ack Required	Bestätigung des Normalzustandes durch den Bediener erforderlich (Quittierung) Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert

Tabelle 264: Parameter für boolesche Ereignisse

10.7.2 Skalare Ereignisse

- Übergänge über Grenzwerte, die für eine skalaren Variable, z. B. einen analogen Eingang, definiert sind.
- Es sind zwei obere und zwei untere Grenzen möglich.
  Für die Grenzwerte muss gelten:
  Oberste Grenze ≥ obere Grenze ≥ Normalbereich ≥ untere Grenze ≥ unterste Grenze.
  Eine Hysterese wird in folgenden Fällen wirken:
  - Bei Unterschreitung einer oberen Grenze
  - Bei Überschreitung einer unteren Grenze
  Durch Angabe einer Hysterese können Sie eine unnötig große Menge an Ereignissen vermeiden, wenn die globale Variable stark um einen Grenzwert schwankt.



1 HH Alarm Value
2 H Alarm Value
3 L Alarm Value
4 LL Alarm Value
5 Hysterese

Bild 81: Fünf Bereiche eines skalaren Ereignisses

Die Parameter der **skalaren** Ereignisse geben Sie im Alarm & Event Editor der Ressource ein, der die folgenden Spalten enthält:

Spalte	Beschreibung	Wertebereich
Name	Name der Ereignisdefinition	Text, max. 31 Zeichen
Globale Variable	Name der zugewiesenen globalen Variable (z. B. eingefügt durch Drag&Drop)	
Datentyp	Datentyp der globalen Variable, nicht änderbar.	abhängig vom Typ der globalen Variablen
Eventquelle	CPU Event: Der Zeitstempel wird auf einem Prozessormodul gebildet. Dieses führt die Ereignisbildung komplett in jedem seiner Zyklen durch. IO Event: Der Zeitstempel wird auf einem geeigneten E/A-Modul gebildet (z. B. AI 32 02). Auto Event: Es wird ein CPU Event und wenn vorhanden IO Events der E/A Module gebildet. Standardwert: CPU Event	CPU Event, IO Event, Auto Event
HH Alarm Text	Text, der den Alarmzustand des obersten Grenzwerts benennt	Text
HH Alarm Value	Oberster Grenzwert, der ein Ereignis auslöst. Bedingung: (HH Alarm Value - Hysterese) > H Alarm Value oder HH Alarm Value = H Alarm Value	abhängig vom Typ der globalen Variablen
HHAlarm Severity	Priorität des obersten Grenzwerts, Standardwert: 1	1...1000
HH Ack Required	Aktiviert: Bediener muss Überschreitung des obersten Grenzwerts bestätigen (Quittierung). Deaktiviert: Bediener muss Überschreitung des obersten Grenzwerts nicht bestätigen. Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
H Alarm Text	Text, der den Alarmzustand des oberen Grenzwerts benennt	Text
H Alarm Value	Oberer Grenzwert, der ein Ereignis auslöst. Bedingung: (H Alarm Value - Hysterese) > (L Alarm Value + Hysterese) oder H Alarm Value = L Alarm Value	abhängig vom Typ der globalen Variablen
HAlarm Severity	Priorität des oberen Grenzwerts, Standardwert: 1	1...1000
H Ack Required	Aktiviert: Bediener muss Überschreitung des oberen Grenzwerts bestätigen (Quittierung). Deaktiviert: Bediener muss Überschreitung des oberen Grenzwerts nicht bestätigen. Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
Return to Normal Text	Text, der den Alarmzustand benennt	Text
Return to Normal Severity	Priorität des Normalzustands, Standardwert: 1	1...1000
Return to Normal Ack Required	Bestätigung des Normalzustandes durch den Bediener erforderlich (Quittierung), Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
L Alarm Text	Text, der den Alarmzustand des unteren Grenzwerts benennt	Text
L Alarm Value	Unterer Grenzwert, der ein Ereignis auslöst. Bedingung: (L Alarm Value + Hysterese) < (H Alarm Value - Hysterese) oder L Alarm Value = H Alarm Value	abhängig vom Typ der globalen Variablen

L Alarm Severity	Priorität des unteren Grenzwerts, Standardwert: 1	1...1000
L Ack Required	Aktiviert Bediener muss Unterschreitung des unteren Grenzwerts bestätigen (Quittierung). Deaktiviert Bediener muss Unterschreitung des unteren Grenzwerts nicht bestätigen. Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
LL Alarm Text	Text, der den Alarmzustand des untersten Grenzwerts benennt	Text
LL Alarm Value	Unterster Grenzwert, der ein Ereignis auslöst. Bedingung: (LL Alarm Value + Hysterese) < (L Alarm Value) oder LL Alarm Value = L Alarm Value	abhängig vom Typ der globalen Variablen
LL Alarm Severity	Priorität des untersten Grenzwerts, Standardwert: 1	1...1000
LL Ack Required	Aktiviert Bediener muss Unterschreitung des untersten Grenzwerts bestätigen (Quittierung). Deaktiviert Bediener muss Unterschreitung des untersten Grenzwerts nicht bestätigen. Standardwert: deaktiviert	Kontrollkästchen aktiviert, deaktiviert
Alarm Hysterese	Die Hysterese verhindert ein ständiges Erzeugen von vielen Ereignissen, wenn der Prozesswert häufig um einen Grenzwert schwankt.	abhängig vom Typ der globalen Variablen

Tabelle 265: Parameter für skalare Ereignisse

10.8 Parameter der X-OPC Server Eigenschaften

i Die gesamte Konfiguration und die Bedienung des X-OPC-Servers führen Sie in SILworX durch. Im SILworX Control Panel können Sie den X-OPC-Server wie eine Steuerung laden, starten und stoppen.

10.8.1 OPC-Server-Set

Das OPC-Server-Set dient als gemeinsame Parametrierbasis für bis zu zwei OPC-Server.

Die Eigenschaften des OPC-Server-Set sind automatisch für beide redundanten X-OPC-Server identisch.

So legen Sie einen neuen OPC-Server-Set an:

1. Öffnen Sie im Strukturbaum **Konfiguration**.
2. Wählen Sie im Kontextmenü der Konfiguration **Neu**, **OPC-Server-Set**, um einen neuen OPC-Server-Set hinzuzufügen.
3. Übernehmen Sie die Standardwerte im Kontextmenü von OPC-Server-Set **Eigenschaften**.

Das Dialogfenster Eigenschaften des OPC-Server-Set enthält die folgenden Parameter.

Element	Beschreibung
Name	Name des OPC-Server-Set. Maximal 31 Zeichen.
Sicherheitszeit [ms]	Die Sicherheitszeit ist die Zeit in Millisekunden, innerhalb welcher der X-OPC-Server auf einen Fehler reagieren muss. Bedingung: Sicherheitszeit ≥ 2 x Watchdog-Zeit Wertebereich: 2000...400 000 ms Standardwert: 20 000 ms
Watchdog-Zeit [ms]	Die Watchdog-Zeit ist die Zeit in Millisekunden, die der X-OPC-Server zum Ausführen eines Programmzyklus maximal benötigen darf. Wird die vorgegebene Watchdog-Zeit überschritten (Ausführung eines Programmzyklus dauert zu lange), wird der X-OPC-Server beendet. Bedingung: WDZ ≥ 1000 ms und ≤ 0.5 * Sicherheitszeit Wertebereich: 1000...200 000 ms Standardwert: 10 000 ms

Hauptfreigabe	Die Einstellung des OPC-Schalters Hauptfreigabe beeinflusst die Funktion der anderen OPC-Schalter. Wenn die Hauptfreigabe ausgeschaltet ist, können die Einstellungen der anderen OPC-Schalter nicht verändert werden, während das Anwenderprogramm abgearbeitet wird (Steuerung in RUN). Standardwert: aktiviert
Autostart	Autostart legt fest, ob die OPC-Konfigurationen nach dem Einschalten oder nach dem Booten des OPC-Servers per Kaltstart, Warmstart oder nicht (Aus) automatisch gestartet werden dürfen. Wenn Autostart deaktiviert ist, geht der X-OPC-Server nach dem Booten in den Zustand STOPP/GÜLTIGE KONFIGURATION. Standardwert: deaktiviert
Start erlaubt	Nur wenn *Start erlaubt* aktiviert ist, kann ein X-OPC-Server vom Programmiergerät aus gestartet werden. Ist *Start erlaubt* deaktiviert, können Sie den X-OPC-Server nicht vom Programmiergerät aus starten. In diesem Fall können Sie den X-OPC-Server nur starten, wenn *Autostart* aktiviert ist und Sie den Host-PC einschalten oder neu booten. Ist weder *Autostart* noch *Start erlaubt* aktiviert, kann der X-OPC-Server nicht mehr starten. Dies kann z. B. bei Wartungsarbeiten erforderlich sein, um das Anlaufen einer Anlage zu verhindern. Standardwert: aktiviert
Laden erlaubt	Wenn *Laden erlaubt* deaktiviert ist, kann keine (neue) OPC-Konfiguration in die Steuerung geladen werden. Deaktivieren Sie *Laden erlaubt*, wenn Sie verhindern wollen, dass in dem X-OPC-Server eine geladene OPC-Konfiguration überschrieben werden kann. Standardwert: aktiviert
Reload erlaubt	Noch keine Funktion!
Globales Forcen erlaubt	Nur wenn *Globales Forcen erlaubt* aktiviert ist, können Sie *Globales Forcen Starten*. i Der Force-Editor kann auch dann zum Anzeigen von Variableninhalten aufgerufen werden, wenn *Globales Forcen erlaubt* deaktiviert ist. Standardwert: deaktiviert

Globale Force Timeout Reaktion	Wenn *Globale Force Timeout Reaktion, Ressource stoppen* eingestellt ist, geht der X-OPC-Server nach Ablauf der voreingestellten Force-Zeit in den Zustand STOPP. Alle Ausgänge des X-OPC-Servers werden auf LOW gesetzt. Wenn *Globale Force Timeout Reaktion, Nur Forcen beenden* eingestellt ist, setzt der X-OPC-Server die Ausführung der OPC-Konfiguration fort, auch nachdem die Force-Zeit abgelaufen ist. i Falls Forcen erlaubt ist, prüfen Sie die Einstellung für Stoppen bei Force-Timeout sorgfältig. Beachten Sie dazu auch die Anmerkungen im Sicherheitshandbuch. Standardwert: Ressource stoppen
Max. Kom.-Zeitscheibe ASYNC [ms]	Die *max. Kom.-Zeitscheibe ASYNC [ms]* ist die Zeit in Millisekunden, die pro X-OPC-Server-Zyklus reserviert wird, um alle anstehenden Kommunikationsaufgaben für die Peer-to-Peer Kommunikation abzuarbeiten. Standardwert: 500 ms
Sollzykluszeit [ms]	Sollzykluszeit des X-OPC-Servers Standardwert: 50 ms
Safeethernet-CRC	
Namensraumtrennzeichen	Punkt . Slash / Doppelpunkt : Backslash \ Standardwert: Punkt
Namensraumtyp	Je nach Anforderung des OPC-Clients kann der Namensraumtyp eingestellt werden: ▪ Hierarchischer Namensraum ▪ Flacher Namensraum Standardwert: Hierarchischer Namensraum
Short tag names for DA	Nur wenn *Flacher Namenraum* gewählt wurde, kann dieser Parameter aktiviert werden. Ist eine Option, bei der Daten und Events ohne weiteren Kontext (Pfadname) dem OPC-Client angeboten werden. Standardwert: deaktiviert
Changeless update	Einstellung je nach Anforderung des OPC-Clients. Aktiviert: Ist *Changeless update* aktiviert, liefert der X-OPC-Server nach Ablauf der OPC Group-UpdateRate immer alle Items zum OPC-Client. Deaktiviert: Ist *Changeless update* deaktiviert, werden nur geänderte Werte dem OPC-Client geliefert (dieses Verhalten entspricht der OPC-Spezifikation).

Zyklusverzögerung [ms]	Die Zyklusverzögerung begrenzt die CPU-Auslastung des PCs durch den X-OPC-Server, damit auch andere Programme noch zur Abarbeitung kommen. Wertebereich: 1 ... 100 ms Standardwert: 5 ms
Simple-Events für CPU-E/A Events	Nie Nur beim Start Immer
Short tag names A&E	Nur wenn *Flacher Namenraum* gewählt wurde, kann dieser Parameter aktiviert werden. Ist eine Option, bei der Daten und Eventquellen ohne weiteren Kontext (Pfadname) dem OPC-Client angeboten werden. Standardwert: deaktiviert

Tabelle 266: Eigenschaften

10.8.2 OPC-Server

So legen Sie einen neuen OPC-Server an:

1. Öffnen Sie im Strukturbaum **Konfiguration**, **OPC-Server-Set**.
2. Wählen Sie im Kontextmenü des OPC-Server-Set **Neu**, **OPC-Server**, um einen neuen OPC-Server hinzuzufügen.
3. Wählen Sie im Kontextmenü von OPC-Server **Eigenschaften**.

Das Dialogfenster Eigenschaften des OPC-Server enthält die folgenden Parameter.

Element	Beschreibung
Name	Name für den OPC-Server. Maximal 31 Zeichen.
System ID [SRS]	Standardwert: 60000

Tabelle 267: Eigenschaften

So öffnen Sie den OPC-Host:

1. Öffnen Sie im Strukturbaum **Konfiguration**, **OPC-Server-Set, OPC-Server**.
2. Wählen Sie im Kontextmenü des OPC-Host **Edit**, um die Übersicht der IP-Schnittstellen zu öffnen.

Das Dialogfenster Edit des OPC-Host enthält die folgenden Parameter.

Element	Beschreibung
PADT-Port	Standardwert: 25138
Name	Name für den OPC-Server-Set. Maximal 31 Zeichen.
IP-Adresse	IP-Adresse des Host PC. Standardwert: 192.168.0.1
Standard-Schnittstelle	Muss aktiviert werden, wenn der Host-PC mehr als einen Ethernet-Port besitzt. Standardwert: aktiviert
HH-Port	Standardwert: 15138

Tabelle 268: Edit

10.9 Deinstallation des X-OPC-Servers

So deinstallieren Sie den X-OPC-Server:

1. Öffnen Sie in Windows **Start, Einstellungen, Systemsteuerung, Software**.
2. Wählen Sie in der Liste den X-OPC-Server aus den Sie deinstallieren wollen und klicken Sie auf **entfernen**.
3. Folgen Sie den Anweisungen der Deinstallationsroutine.

11 ComUserTask

Neben dem Logikprogramm, das mit SILworX erstellt wird, können Sie zusätzlich ein C-Programm auf der Steuerung betreiben.

Dieses nicht sichere C-Programm läuft als ComUserTask rückwirkungsfrei zum sicheren Prozessormodul auf dem Kommunikationsmodul der Steuerung.

Die ComUserTask hat einen eigenen Zyklus, der unabhängig von dem Zyklus der CPU ist.

Damit können Sie beliebige Anwendungen in C erstellen und als ComUserTask implementieren z.B.:

- Kommunikationsschnittstellen für spezielle Protokolle (TCP, UDP usw.).
- Gateway Funktion zwischen TCP/UDP und serieller Kommunikation.

11.1 Systemanforderung

Benötigte Ausstattung und Systemanforderung:

Element	Beschreibung
Steuerung	HIMax mit COM-Modul
CPU-Modul	Die Ethernet-Schnittstellen des Prozessormoduls können für ComUserTask nicht verwendet werden.
COM-Modul	Ethernet 10/100BaseT D-SUB Anschlüsse FB1 und FB2 z.B. für RS232 Werden die seriellen Feldbusschnittstelle (FB1 oder FB2) verwendet, müssen diese mit einem optionalen HIMA Submodul ausgerüstet sein, siehe Kapitel 3.4.
Aktivierung	Die Freischaltung erfolgt per Software-Freischaltcode, siehe Kapitel 3.5.

Tabelle 269: Systemanforderung und Ausstattung ComUserTask

Eigenschaften der ComUserTask:

Element	Beschreibung
ComUserTask	Es kann für jede HIMax Steuerung eine ComUserTask konfiguriert werden.
Sicherheitsgerichtet	Nein
Datenaustausch	Konfigurierbar
Code- und Datenbereich	Startadresse 0x790000 Länge 448 kByte
Stack	Der Stack liegt in einem dafür reservierten Speicher außerhalb des Code-/Datenbereichs. Länge 64kByte

Tabelle 270: Eigenschaften ComUserTask

11.1.1 Anlegen einer ComUserTask

So legen Sie eine neue ComUserTask an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **ComUserTask** wählen, um eine neue ComUserTask hinzuzufügen.
3. Im Kontextmenü der ComUserTask **Eigenschaften** das **COM-Modul** auswählen. Standardeinstellungen können für die erste Konfiguration beibehalten werden.

11.2 Voraussetzungen

Für die Programmierung der ComUserTask steht zusätzlich zum normalen Befehlsumfang von C eine eigene Library (siehe Kapitel 11.4) mit definierten Funktionen zur Verfügung.

Entwicklungsumgebung

Zu der Entwicklungsumgebung gehören der GNU C Compiler und Cygwin, die auf einer gesonderten Installations-CD vorhanden sind (nicht bei SILworX mit dabei) und den Bedingungen der GNU General Public License unterliegen (siehe www.gnu.org).

Die neuesten Versionen und Dokumentationen zu der Entwicklungsumgebung können Sie sich von den entsprechenden Seiten im Internet herunterladen www.cygwin.com und www.gnu.org.

Steuerung

Bei den HIMax Steuerungen hat die ComUserTask keinen Zugriff auf die sicheren Hardware Ein- und Ausgänge. Wenn ein Zugriff auf die Hardware Ein- und Ausgänge benötigt wird, dann ist ein Anwenderprogramm der CPU zur Verbindung der Variablen erforderlich (siehe 11.4.5).

11.3 Abkürzungen

Abkürzung	Bedeutung
CUCB	COM User Callback (CUCB_ Funktionen werden von der COM aufgerufen)
CUIT	COM User IRQ Task
CUL	COM User Library (CUL_ Funktionen werden in der CUT aufgerufen)
CUT	ComUserTask
GNU	GNU Projekt
IF	InterFace
FB	Feldbusschnittstelle der Steuerung
FIFO	First In First Out (Datenspeicher)
NVRam	Non Volatile Random Access Memory, nicht flüchtiger Speicher

Tabelle 271: Abkürzungen

11.4 CUT-Schnittstelle in SILworX

Die Prozessdatenkommunikation der ComUserTask läuft zwischen der COM und der CPU ab.

⚠ WARNUNG

Die geladene ComUserTask darf keine privilegierten Befehle des COM Moduls benutzen.

Der Code der CUT läuft auf der COM rückwirkungsfrei zur CPU. Damit ist die sichere CPU vor dem Code der CUT geschützt.
Es ist jedoch zu beachten, dass Fehler im CUT Code die Funktion der COM als Ganzes und damit auch die Funktion der Steuerung stören und sogar unterbinden können.
Die Sicherheitsfunktionen der CPU werden dadurch nicht beeinträchtigt.

11.4.1 Schedule-Intervall [ms]

Die ComUserTask wird in einem parametrierten Schedule-Intervall[ms] in den Zuständen RUN und STOP_VALID_CONFIG der Steuerung (COM- Modul) aufgerufen.

Das Schedule-Intervall[ms] wird in SILworX in den *Eigenschaften* der ComUserTask eingestellt.

Schedule-Intervall[ms]	
Wertebereich:	10 .. 255 ms
Standardwert:	15 ms

Tabelle 272: Schedule-Intervall [ms]

i Die der CUT verfügbare COM-Prozessor-Zeit hängt von den anderen parametrierten Funktionen der COM, wie z.B. safe**ethernet**, Modbus-TCP etc. ab.

Wenn die CUT nicht innerhalb des Schedule-Intervalls fertig wird, dann wird jeder Aufruf zum Neustart der CUT abgewiesen, bis die CUT abgearbeitet ist.

11.4.2 Scheduling-Vorlauf

Im Betriebsmode RUN der Steuerung:

Vor jedem Aufruf der CUT stellt die COM die Prozessdaten von der sicheren CPU der CUT in einem von der CUT definierten Speicherbereich zur Verfügung.

Im Betriebsmodus STOP der Steuerung:

Es findet kein Prozessdatenaustausch von der COM zur sicheren CPU statt.

11.4.3 Scheduling-Nachlauf

Im Betriebsmode RUN der Steuerung:

Nach jedem Aufruf der CUT stellt die COM die Prozessdaten von der CUT der sicheren CPU zur Verfügung.

Im Betriebsmodus STOP der Steuerung:

Es findet kein Prozessdatenaustausch von der COM zur sicheren CPU statt.

11.4.4 STOP_INVALID_CONFIG

Befindet sich die COM Im Zustand STOP_INVALID_CONFIG, dann wird die CUT nicht ausgeführt.

Wechselt die COM in den Zustand STOP_INVALID_CONFIG und führt die CUT oder die CUIT aus, so werden diese terminiert.

11.4.5 Variablen der CUT-Schnittstelle (CPU<->CUT)

Parametrierung einer nicht sicherheitsgerichtete Prozessdatenkommunikation zwischen sicherer CPU und COM (CUT).

Senderichtung	Maximale Größe der Prozessdaten
COM->CPU	16375 Bytes Daten (16384 Bytes – 9 Statusbytes)
CPU->COM	16382 Bytes Daten (16384 Bytes – 2 Steuerungsbytes)

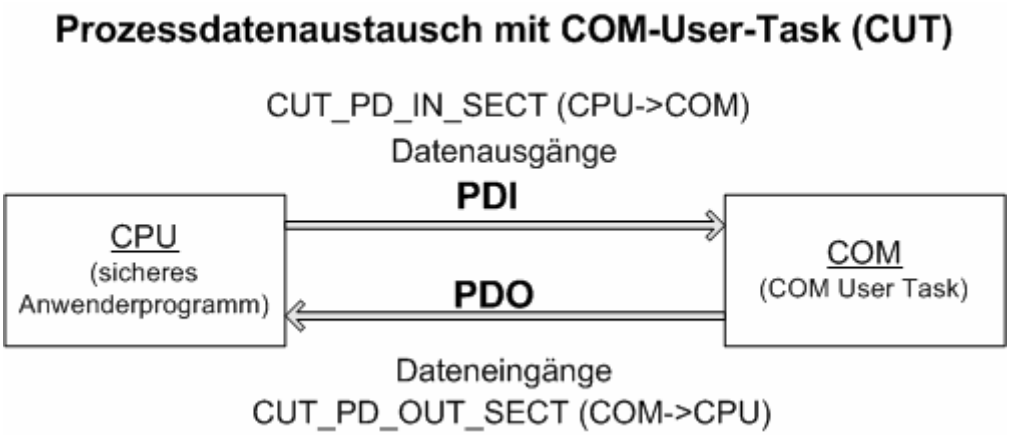


Bild 82: Prozessdatenaustausch zwischen CPU und COM (CUT)

Es können alle Datentypen ausgetauscht werden, die in SILworX verwendet werden.

Die Struktur der Daten muss in SILworX parametriert werden.
Die Größe der Datenstrukturen CUT_PDI und CUT_PDO (im compilerten C-Code der CUT) müssen der gleichen Größe der konfigurierten Datenstruktur in SILworX entsprechen.

i Sind im compilierten C-Code die Datenstrukturen CUT_PDI und CUT_PDO nicht vorhanden oder haben nicht die gleiche Größe wie die Datenstruktur in SILworX parametrierten Prozessdaten, dann ist die Konfiguration ungültig und die COM nimmt den Zustand STOP_INVALID_CONFIG an.

Die Prozessdatenkommunikation findet nur im Betriebsmodus RUN statt.

Über die Menüfunktion **Edit** erreichen Sie die Register Prozessvariablen und Systemvariablen.

Systemvariablen

Das Register **Systemvariablen** enthält die folgenden Systemparameter zur Überwachung und Steuerung des CUT:

<table>
<tr><th>Name</th><th>Funktion</th></tr>
<tr><td>Ausführungszeit
[DWORD]</td><td>Ausführungszeit der ComUserTask in µs</td></tr>
<tr><td>Reales Schedule-Intervall
[DWORD]</td><td>Zeitabstand zwischen zwei ComUserTask Durchläufen in ms</td></tr>
<tr><td>Steuerung des User Task-Zustands
[WORD]</td><td>Die folgende Tabelle zeigt die Möglichkeiten, wie der Anwender mit dem Systemparameter Steuerung des User Task-Zustandes die ComUserTask steuern kann:
<table>
<tr><th>Funktion</th><th>Beschreibung</th></tr>
<tr><td>DISABLED
0x8000</td><td>Das Anwenderprogramm sperrt die CUT (d.h. die CUT wird nicht gestartet).</td></tr>
<tr><td>AUTOSTART
0 (Default)</td><td>Nach Terminierung der CUT startet die CUT automatisch nachdem die Störung oder der Fehler beseitigt wurde.</td></tr>
<tr><td>TOGGLE_MODE_0
0x0100</td><td>Nach Terminierung der CUT ist ein Start der CUT erst wieder nach dem Setzen von TOGGLE_MODE_1 erlaubt.</td></tr>
<tr><td>TOGGLE_MODE_1
0x0101</td><td>Nach Terminierung der CUT ist ein Start der CUT erst wieder nach dem Setzen von TOGGLE_MODE_0 erlaubt.</td></tr>
</table></td></tr>
<tr><td>Zustand der User Task
[BYTE]</td><td>1 = RUNNING (CUT läuft)
0 = ERROR (CUT läuft nicht wegen eines Fehlers)</td></tr>
</table>

Tabelle 273: Systemvariablen der ComUserTask

Prozessvariablen

Eingangssignale (COM->CPU)

In das Register **Eingangssignale** werden die Vaiablen eingetragen, die von der COM (CUT) zur CPU übertragen werden sollen (Eingangsbereich der CPU).

⚠ VORSICHT

Unsichere Daten der ComUserTask!

Die nicht sicheren Variablen der ComUserTask dürfen die Sicherheitsfunktionen des CPU Anwenderprogramms nicht behindern.

Eingangssignale des ComUserTask

Name	Datentyp	Offset
CUT_Zaehler	DWORD	0
Time_Stamp	DWORD	4

Tabelle 274: Eingangssignale des ComUserTask

Erforderlicher Eintrag im C-Code

Der C-Code des ComUserTasks muss für die Ausgänge der COM (Eingangsbereich der CPU) die folgende Datenstruktur CUT_PDO enthalten:

```
/* SILworX Input Records (COM->CPU) */
uword CUT_PDO[1] __attribute__ ((section("CUT_PD_OUT_SECT"), aligned(1)));
```

Die Größe der Datenstruktur CUT_PDO muss der Größe der konfigurierten Daten-Eingänge in SILworX entsprechen.

Ausgangssignale (CPU->COM)

In das Register **Ausgangssignale** werden die Variablen eingetragen, die von der CPU (Ausgangsbereich der CPU) zur COM (CUT) übertragen werden sollen.

Ausgangssignale des ComUserTask

Name	Datentyp	Offset
CPU_COM_1	WORD	0
CPU_COM_2	WORD	2

Tabelle 275: Ausgangssignale des ComUserTask

Erforderlicher Eintrag im C-Code

Der C-Code des ComUserTasks muss für die Eingänge der COM (Ausgangsbereich der CPU) die folgende Datenstruktur CUT_PDI enthalten:

```
/* SILworX Output Records (CPU->COM) */
uword CUT_PDI[1] __attribute__ ((section("CUT_PD_IN_SECT"), aligned(1)));
```

Die Größe der Datenstruktur CUT_PDI muss der Größe der konfigurierten Daten-Ausgänge in SILworX entsprechen.

11.5 CUT Funktionen

11.5.1 COM-User-Callback-Funktionen

Die COM-User-Callback Funktionen haben alle den Prefix **CUCB_** und werden bei Ereignissen direkt von der COM aufgerufen.

i Alle COM-User-Callback Funktionen müssen im C-Code des Anwenders definiert werden!
Auch die Funktion CUCB_IrqService die für HIMax und HIMatrix nicht unterstützt wird, muss im C-Code des Anwenders aus Kompatibilitätsgründen definiert werden.
Funktionsprototyp: void CUCB_IrqService(udword devNo) {}

Die COM-User-Callback (CUCB) und die COM-User-Library (CUL) Funktionen teilen sich den gleichen Code- und Datenspeicher sowie den Stack. Diese Funktionen stellen gegenseitig die Konsistenz der gemeinsam verwendeten Daten (Variablen) sicher.

11.5.2 COM-User-Library-Funktionen

Alle COM-User-Library-Funktionen und Variablen haben den Prefix **CUL_** und werden in der CUT aufgerufen.

Diese CUL Funktionen sind alle über das Objekt-File **libcut.a** verfügbar.

11.5.3 Header-Files

Die beiden Header Files **cut.h** und **cut_types.h** enthalten alle Funktionsprototypen für CUL/CUCB und die zugehörigen Datentypen und Konstanten.

Zur verkürzenden Schreibweise werden die folgenden Datentypen im Header-File **cut_types.h** definiert:

```
typedef unsigned long     udword;
typedef unsigned short    uword;
typedef unsigned char     ubyte;
typedef signed long       dword;
typedef signed short      word;
typedef signed char       sbyte;
#ifndef HAS_BOOL
typedef unsigned char     bool; // mit 0=FALSE, sonst TRUE
#endif
```

11.5.4 Code-/Datenbereich und Stack für die CUT

Der Code-/Datenbereich ist ein zusammenhängender Speicherbereich, der mit dem Code-Segment und dem Initialdaten-Segment beginnt und mit den Datensegmenten fortgesetzt wird. Im HIMA Linkersteuerfile (**makeinc.inc.app** und **section.dld**) ist die beschriebene Reihenfolge der Segmente und die verfügbare Speichermenge festgelegt (Gilt für HIMax und HIMatrix).

Die ComUserTask teilt mithilfe des HIMA Linkersteuerfiles den verfügbaren Speicherbereich optimal zwischen dem Code und den Daten auf.

Startadresse 0x790000

Länge 448kByte

Der Stack liegt in einem reservierten Speicherbereich, der zur Laufzeit des COM-Betriebssystems festgelegt wird.

Endadresse Dynamisch aus Sicht der CUT

Länge 64kByte

11.5.5 Startfunktion CUCB_TaskLoop

Die Funktion `CUCB_TaskLoop()` ist die Startfunktion zur ComUserTask.

Die Programmausführung der ComUserTask beginnt mit dem Aufruf dieser Funktion (siehe `Schedule-Intervall[ms]` Kapitel 11.4.1).

Funktionsprototyp:

void CUCB_TaskLoop(udword mode)

Parameter:

Die Funktion hat den folgenden Parameter

Parameter	Beschreibung
mode	1 = MODE_STOP entspricht dem Modus STOP_VALID_CONFIG 2 = MODE_RUN normaler Betrieb der Steuerung

Tabelle 276: Parameter

11.5.6 Serielle Schnittstellen RS485 / RS232 IF

Die verwendeten Feldbusschnittstellen müssen mit den entsprechendes Feldbus Submodul (Hardware) bestückt sein.

---

i Für jede HIMax Steuerung steht die jeweilige Systemdokumentationen zur Verfügung.

CUL_AscOpen

Die Funktion `CUL_AscOpen()` initialisiert die eingegebene serielle Schnittstelle (*comId*) mit den übergebenen Parametern. Nach dem Aufruf der Funktion `CUL_AscOpen()` beginnt die COM sofort mit dem Empfang von Daten über diese Schnittstelle.

Die empfangenen Daten werden in einem Software FIFO der Größe 1kByte für **jede** initialisierte serielle Schnittstelle gespeichert.

Die Daten werden solange gespeichert, bis diese mit der Funktion `CUL_AscRcv()`von der CUT ausgelesen werden.

---

i Ist das Auslesen der Daten aus dem FIFO langsamer als der Empfang neuer Daten, so werden die neu empfangenen Daten verworfen.

Funktionsprototyp:

```
udword CUL_AscOpen(  Udword comId,
                     Ubyte duplex,
                     udword baudRate,
                     ubyte parity,
                     ubyte stopBits)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE
duplex	0 = Full-Duplex (nur für FB4 falls RS232 erlaubt) 1 = Halb-Duplex
baudRate	1 = 1200 Bit 2 = 2400 Bit 3 = 4800 Bit 4 = 9600 Bit 5 = 19200 Bit 6 = 38400 Bit 7 = 57600 Bit 8 = 115000 Bit
Die Datenbitlänge ist fest auf 8 Datenbits eingestellt. Zu diesen 8 Datenbits kommen die parametrierten Bits für Parity und Stopbits, sowie ein Startbit hinzu.	
parity	0 = NONE 1 = EVEN 2 = ODD
stopBits	1 = 1 Bit 2 = 2 Bits

Tabelle 277: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-Datei cut.h definiert.

Error code	Beschreibung
CUL_OKAY	Die Initialisierung wurde erfolgreich ausgeführt.
CUL_ALREADY_IN_USE	Die Schnittstelle wird durch andere Funktionen der COM benutzt oder ist bereits offen.
CUL_INVALID_PARAM	Es wurden unzulässige Parameter oder Parameterkombinationen übergeben.
CUL_DEVICE_ERROR	Sonstige Fehler

Tabelle 278: Rückgabewert

CUL_AscClose

Die Funktion `CUL_AscClose()`schließt die in *comId* eingetragene serielle Schnittstelle. Dabei werden die bereits empfangenen, aber noch nicht mit der Funktion `CUL_AscRcv()`ausgelesenen Daten im FIFO gelöscht.

Funktionsprototyp:

Udword CUL_AscClose(udword comId)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE

Tabelle 279: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
CUL_OKAY	Die Schnittstelle wurde erfolgreich geschlossen.
CUL_NOT_OPENED	Die Schnittstelle war nicht (durch die CUT) geöffnet.
CUL_INVALID_PARAM	Es wurden unzulässige Parameter oder Parameterkombinationen übergeben.
CUL_DEVICE_ERROR	Sonstige Fehler

Tabelle 280: Rückgabewert

CUL_AscRcv

Die Funktion `CUL_AscRcv()`beauftragt die COM, eine definierte Datenmenge aus dem FIFO zur Verfügung zu stellen.

Sobald die angefragte Datenmenge verfügbar ist (und die CUL bzw. das Scheduling es zulassen), ruft die COM die Funktion *CUCB_AscRcvReady()* auf.

Sind nicht genug Daten im FIFO, so kehrt die Funktion `CUL_AscRcv()` sofort zurück.

Der Auftrag für den Datenempfang bleibt solange gespeichert bis:

- Der Auftrag vollständig abgearbeitet wurde oder
- die Funktion *CUL_AscClose()* aufgerufen wird oder
- durch einen neuen Auftrag neu definiert wird.

i Bis der Auftrag fertig ist, darf der Inhalt von *pBuf nur noch über die Funktion *CUCB_AscRcvReady()* geändert werden.

Funktionsprototyp:

Udword CUL_AscRcv(udword comId, CUCB_ASC_BUFFER *pBuf)

```
typedef struct CUCB_AscBuffer {

      bool  bAscState;  // zur Verwendung durch CUT/CUCB
      bool  bError;     // zur Verwendung durch CUT/CUCB
      uword align;      // COM ist 4 aligned, long's sind performanter
      udword mDataIdx;  // Byte-Offset in aData, ab dem die Daten liegen
      udword mDataMax;  // max. Byte-Offset: (mDataMax-mDataIdx) gibt an,
                        // wieviele Bytes in aData gesendet oder
                        // empfangen werden müssen
      udword aData[1];  // Beginn des Datenkopierbereichs
      }CUCB_ASC_BUFFER;
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE
pBuf	Definiert die angeforderte Datenmenge und den Ort, an den sie kopiert werden soll, bevor dann CUCB_AscReady() aufgerufen wird. Sind bereits ausreichend Daten im FIFO vorhanden, wird während CUL_AscRcv() CUCB_AscRcvReady() aufgerufen.

Tabelle 281: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
CUL_OKAY	wenn der Auftrag erfolgreich war, sonst Fehlercode.
CUL_NOT_OPENED	falls die Schnittstelle nicht durch die CUT geöffnet wurde
CUL_INVALID_PARAM	Es wurden unzulässige Parameter oder Parameterkombinationen übergeben.
CUL_DEVICE_ERROR	Sonstige Fehler

Tabelle 282: Rückgabewert

Restriktionen:

- Falls der durch CUCB_ASC_BUFFER definierte Speicherbereich nicht im Datensegement der CUT liegt, werden die CUIT und die CUT terminiert.
- Es können maximal 1024 Byte Daten angefordert werden.

CUCB_AscRcvReady

Wenn die COM die Funktion `CUCB_AscRcvReady()` aufruft, dann liegt die angeforderte Datenmenge im FIFO bereit (Daten von der im Parameter `comId` definierten seriellen Schnittstelle).

Die Daten wurde zuvor mit der Funktion `CUL_AscRcv()`angefordert.

Der Aufruf der Funktion `CUCB_AscRcvReady()` kann außerhalb und während des Aufrufs der Funktion `CUL_AscRcv()` erfolgen. Der Task-Kontext ist immer der der CUT.

Die Funktion `CUCB_AscRcvReady()` darf alle CUT-Library-Funktionen aufrufen.

Ebenfalls erlaubt ist

- die Erhöhung von `mDataMax`, bzw.
- die neue Parametrierung von `mDataIdx` und `mDataMax` von der `comId` zugeordneten `*pBuf` Daten (zum Weiterlesen).

Das Strukturelement von CUCB_ASC_BUFFER.`mDataIdx` hat den Wert von CUCB_ASC_BUFFER.`mDataMax`.

Funktionsprototyp:

void CUCB_AscRcvReady(udword comId)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE

Tabelle 283: Parameter

Restriktionen:

Falls der durch CUCB_ASC_BUFFER definierte Speicherbereich nicht im Datensegment von CUT liegt, werden CUIT und CUT terminiert.

CUL_AscSend

Die Funktion CUCB_AscSend sendet die durch den Parameter pBuf definierte Datenmenge über die serielle Schnittstelle comId.

Die definierte Datenmenge muss ≥ 1 Byte und ≤ 1kByte sein.

Nach erfolgtem Senden wird die Funktion CUCB_AscSendReady( )aufgerufen.
Im Fehlerfall wird

- nicht gesendet und
- die Funktion CUCB_AscSendReady( ) wird nicht aufgerufen.

Funktionsprototyp:

Udword CUL_AscSend(udword comId, CUCB_ASC_BUFFER *pBuf)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE
pBuf	Definiert die zu sendende Datenmenge

Tabelle 284: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
CUL_OKAY	falls das Versenden erfolgreich durchgeführt wurde
CUL_WOULDBLOCK	falls eine zuvor versendete Nachricht noch nicht versendet wurde
CUL_NOT_OPENED	falls die Schnittstelle nicht von CUT geöffnet wurde
CUL_INVALID_PARAM	Es wurden unzulässige Parameter oder Parameterkombinationen übergeben.
CUL_DEVICE_ERROR	Sonstige Fehler

Tabelle 285: Rückgabewert

Restriktionen:

Falls der durch CUCB_ASC_BUFFER definierte Speicherbereich nicht im Datensegement von CUT liegt, werden CUIT und CUT terminiert.

CUCB_AscSendReady

Wenn die COM die Funktion `CUCB_AscSendReady()` aufruft, dann ist das Senden der Daten mit der Funktion `CUCB_AscSend()` über die serielle Schnittstelle abgeschlossen.

Der Task-Kontext ist immer der der CUT. Die Funktion `CUCB_AscSendReady()` darf alle CUT-Library-Funktionen aufrufen.

Funktionsprototyp:

void CUCB_AscSendReady(udword comId)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
comId	Feldbusschnittstelle (RS485, RS 232) 1 = FB1 2 = FB2 3 = FB3 4 = FB4_SERVICE

Tabelle 286: Parameter

11.5.7 UDP/TCP-Socket-IF

Maximal 8 Sockets stehen unabhängig vom verwendeten Protokoll zur gleichzeitigen Nutzung zur Verfügung.

Die physikalische Verbindung erfolgt über die 10/100BaseT Ethernet Schnittstellen der Steuerung.

CUL_SocketOpenUdpBind

Die Funktion `CUL_SocketOpenUdpBind()` erzeugt einen Socket vom Typ UDP und bindet den Socket an den ausgewählten Port.

Die Adresse für das Binden ist immer INADDR_ANY, d.h. alle an die COM adressierten Nachrichten für UDP/port werden empfangen. Sockets werden immer im non-blocking Mode betrieben; d.h. diese Funktion blockiert nicht.

Funktionsprototyp:

dword CUL_SocketOpenUdpBind(uword port, uword *assigned_port_ptr)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
port	Eine freie, durch die COM nicht belegte Portnummer >= 0. Ist der Parameter port = 0, dann wird der Socket an den ersten freien Port gebunden.
assigned_port_ptr	Adresse, an die die gebundene Portnummer kopiert werden soll, falls port = 0 ist, oder NULL falls nicht

Tabelle 287: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
socketNummer	Vergebene SocketNummer für UDP falls > 0; Fehlercodes sind < 0
CUL_ALREADY_BOUND	Binden an port für UDP nicht möglich
CUL_NO_MORE_SOCKETS	Keine Ressourcen für Socket mehr verfügbar
CUL_SOCK_ERROR	Andere Socket Fehler

Tabelle 288: Rückgabewert

Restriktionen:

Ist assigned_port_ptr nicht im Besitz der CUT, so werden CUT/CUIT terminiert.

CUL_SocketOpenUdp

Die Funktion `CUL_SocketOpenUdp()` erzeugt einen Socket vom Typ UDP ohne Anbindung an einen Port. Danach können die Nachrichten über den Socket nur versendet werden, kein Empfang.

Funktionsprototyp:

dword CUL_SocketOpenUdp (void)

Parameter:

Keine

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
socketNummer	Vergebene SocketNummer für UDP falls > 0 Fehlercodes sind < 0
CUL_NO_MORE_SOCKETS	Keine Ressourcen für Socket mehr verfügbar
CUL_SOCK_ERROR	Andere Socket Fehler

Tabelle 289: Rückgabewert

CUL_NetMessageAlloc

Die Funktion `CULMessageAlloc()` alloziert Messagespeicher für die Nutzung von

- `CUL_SocketSendTo()` bei UDP und
- `CUL_SocketSend()` bei TCP

Es können maximal 10 Messages gleichzeitig in der CUT in Benutzung sein.

Funktionsprototyp:

void *CUL_NetMessageAlloc(udword size, ubyte proto)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
size	Benötigte Speichermenge in Bytes, muss ≥ 1 Byte und ≤ 1400 Byte sein
proto	0 = TCP 1 = UDP

Tabelle 290: Parameter

Rückgabe:

Puffer Adresse, an die die zu sendenden Nutzdaten kopiert werden müssen. Es dürfen niemals Speicherbereiche außerhalb des allozierten Bereichs beschrieben werden. Es stehen keine Bereiche für die verwendeten Transportprotokolle zur Verfügung (Ethernet/IP/UDP oder TCP).

Restriktionen:

Falls keine Speicherressourcen mehr zur Verfügung stehen oder die Parametergröße zu groß oder proto > 1 ist, werden die CUT und die CUIT terminiert.

CUL_SocketSendTo

Die Funktion `CUL_SocketSendTo()` versendet die zuvor mit `CUL_NetMessageAlloc()` allozierte und gefüllte Nachricht als UDP Paket an die Zieladresse destIp/destPort.

Nach der Sendung wird der Messagespeicher pMsg wieder automatisch freigegeben.

Bei jeder Sendung muss mit der Funktion `CULMessageAlloc()`zuerst Messagespeicher alloziert werden.

Funktionsprototyp:

```
dword CUL_SocketSendTo( dword socket,
                        void *pMsg,
                        udword size,
                        udword destIp,
                        uword destPort)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
Socket	zuvor mit CUL_SocketOpenUdp() erzeugter Socket
pMsg	zuvor mit CUL_NetMessageAlloc() reservierter Speicher der UDP Nutzdaten
Size	Speichermenge in Bytes, muss ≤ der zuvor allozierten Menge sein
destIp	Zieladresse != 0, auch 0xffffffff als Broadcast erlaubt
destPort	Zielport != 0

Tabelle 291: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im headerfile cut.h definiert.

Error code	Beschreibung
CUL_OKAY	Message erfolgreich versendet
CUL_NO_ROUTE	Kein Routing vorhanden um destIp zu erreichen
CUL_WRONG_SOCK	Falscher Socket-Typ oder Socket nicht vorhanden
CUL_SOCK_ERROR	Andere Socket Fehler

Tabelle 292: Rückgabewert

Restriktionen:

Ist pMsg keine im Besitz der CUT befindliche Message oder ist size für pMsg zu groß, so werden CUT/CUIT terminiert.

CUCB_SocketUdpRcv

Die COM ruft die Funktion CUCB_SocketUdpRcv() auf, wenn Daten vom Socket bereit liegen.. Im Callback müssen die Daten bei Bedarf aus `*pMsg` nach CUT-Data kopiert werden. Nach dem `return` der Funktion darf auf `*pMsg` nicht mehr zugegriffen werden.

Funktionsprototyp:

```
void CUCB_SocketUdpRcv( dword socket,
                        void *pMsg,
                        udword packetLength,
                        udword dataLength)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
socket	zuvor mit CUL_SocketOpenUdp() erzeugter Socket
`pMsg`	`pMsg` zeigt auf den Beginn des UDP-Paket inklusive dem Ethernet-Header. Über den Ethernet-Header kann der Sender der Message ermittelt werden.
packetLength	Die Länge des Paketes steht in packetLength, dabei ist die Länge des Headers mit enthalten.
dataLength	Die Länge des UDP Nutzdatenanteils steht in dataLength.

Tabelle 293: Parameter

CUL_NetMessageFree

Die Funktion `CUL_NetMessageFree()` gibt die zuvor mit `CUL_NetMessageAlloc()` allozierten Nachricht frei.

Diese Funktion ist im Normalfall nicht notwendig, da durch den Aufruf der Funktion `CUL_SocketSendTo()` eine automatische Freigabe erfolgt.

Funktionsprototyp:

void CUL_NetMessageFree(void *pMsg)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
`pMsg`	zuvor mit `CUL_NetMessageAlloc()` reservierter Speicher

Tabelle 294: Parameter

Restriktionen:

Ist `pMsg` keine im Besitz der CUT befindliche Message, so werden CUT/CUIT terminiert.

CUL_SocketOpenTcpServer

Die Funktion `CUL_SocketOpenServer()` erzeugt einen Socket vom Typ TCP und bindet den Socket an den ausgewählten Port.

Die Adresse für das Binden ist immer INADDR_ANY. Zusätzlich wird die COM beauftragt auf dem Stream-Socket ein *listen* auszuführen. Sockets werden immer im non-blocking Mode betrieben; d.h. diese Funktion hier blockiert nicht.

Für die weitere Bedienung des Sockets siehe `CUCB_SocketTryAccept()` und `CUL_SocketAccept()`.

Funktionsprototyp:

dword CUL_SocketOpenTcpServer(uword port, udword backlog)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
port	durch die COM nicht belegte Portnummer > 0
backlog	maximale Anzahl wartender Verbindungsaufnahmen für Socket

Tabelle 295: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
socketNummer	Vergebene SocketNummer für UDP falls > 0 Fehlercodes sind < 0
CUL_ALREADY_BOUND	Binden an port/proto nicht möglich
CUL_NO_MORE_SOCKETS	Keine Ressourcen für Socket mehr verfügbar
CUL_SOCK_ERROR	Andere Socket Fehler

Tabelle 296: Rückgabewert

Restriktionen:

Im erfolgreichen Fall wird 1 Socket verbraucht.

CUCB_SocketTryAccept

Die COM ruft die Funktion `CUL_SocketTryAccept()` auf, wenn eine TCP-Verbindungsanfrage ansteht.

Mit dieser Anfrage kann dann mit der Funktion `CUL_SocketAccept()` ein Socket angelegt werden.

Funktionsprototyp:

void CUCB_SocketTryAccept(dword serverSocket)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
serverSocket	zuvor mit *CUL_SocketOpenTcpServer()* erzeugter Socket.

Tabelle 297: Parameter

CUL_SocketAccept

Die Funktion `CUL_SocketAccept()` erzeugt für die zuvor mit *CUCB_SocketTryAccept()* signalisierte Verbindungsanfrage einen neuen Socket.

Funktionsprototyp:

```
dword CUL_SocketAccept( dword serverSocket,
                        udword *pIpAddr,
                        uword *pTcpPort)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
serverSocket	unmittelbar zuvor mit CUCB_SocketTryAccept() signalisierter serverSocket
pIpAddr	Adresse, an die die IP Adresse des Peers kopiert werden soll oder 0 falls nicht
pTcpPort	Adresse, an die die TCP Portnummer des Peers kopiert werden soll oder 0 falls nicht.

Tabelle 298: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
Socket	falls > 0, neu erzeugter Socket
CUL_WRONG_SOCK	Falscher Socket-Typ oder Socket nicht vorhanden
CUL_NO_MORE_SOCKETS	es sind keine Socket-Ressourcen mehr verfügbar
CUL_SOCK_ERROR	andere Socket Fehler

Tabelle 299: Rückgabewert

Restriktionen:

Sind pIpAddr und pTcpPort nicht im Besitz der CUT, so werden CUT/CUIT terminiert.

CUL_SocketOpenTcpClient

Die Funktion `CUL_SocketOpenTcpClient()` erzeugt einen Socket vom Typ TCP mit freiem lokalem Port und beauftragt eine Verbindung zu `destIp` und `destPort`. Sockets werden immer im non-blocking Mode betrieben; d.h. diese Funktion blockiert nicht. Sobald die Verbindung hergestellt wurde, wird `CUCB_SocketConnected()` aufgerufen.

Funktionsprototyp:

dword CUL_SocketOpenTcpClient(udword destIp, uword destPort)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
destIp	IP-Adresse des Kommunikationspartners
destPort	Portnummer des Kommunikationspartners

Tabelle 300: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
socketNummer	falls > 0; Fehlercodes sind < 0
CUL_NO_MORE_SOCKETS	Keine Ressourcen für Socket mehr verfügbar
CUL_NO_ROUTE	kein Routing vorhanden, um destIp zu erreichen
CUL_SOCK_ERROR	andere Socket Fehler

Tabelle 301: Rückgabewert

CUCB_SocketConnected

Die Funktion `CUCB_SocketConnected()` wird von der COM aufgerufen, wenn mit der Funktion `CUL_SocketOpenTcpClient()` eine TCP-Verbindung aufgebaut wurde.

Funktionsprototyp:

void CUCB_SocketConnected(dword socket, bool successfully)

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
socket	zuvor mit `CUL_SocketOpenTcpClient()` erzeugter und beauftragter Socket
successfully	TRUE, falls der Verbindungsversuch erfolgreich war, ansonsten FALSE

Tabelle 302: Parameter

CUL_SocketSend

Die Funktion `CUL_SocketSend()` versendet die zuvor mit `CUL_NetMessageAlloc()` allozierte und gefüllte Nachricht als TCP Paket.

Nach der Sendung wird der Messagespeicher pMsg wieder automatisch freigegeben.

Bei jeder Sendung muss mit der Funktion `CULMessageAlloc()`zuerst Messagespeicher alloziert werden.

Funktionsprototyp:

```
dword CUL_SocketSend(     dword socket,
                          void *pMsg,
                          udword size)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
socket	zuvor mit `CUL_SocketAccept()/CUL_SocketOpenTcpClient()` erzeugter Socket
pMsg	zuvor mit `CUL_NetMessageAlloc()` reservierter Speicher der TCP Nutzdaten
size	Speichermenge in Bytes, muss ≤ der zuvor allozierten Menge sein

Tabelle 303: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind im Header-File cut.h definiert.

Error code	Beschreibung
CUL_OKAY	Message erfolgreich versendet
CUL_WRONG_SOCK	Falscher Socket-Typ oder Socket nicht vorhanden
CUL_WOULD_BLOCK	Message kann nicht versendet werden, da sonst Socket blockieren würde
CUL_SOCK_ERROR	andere Socket Fehler

Tabelle 304: Rückgabewert

Restriktionen:

Ist `pMsg` keine im Besitz der CUT befindliche Message oder ist size für `pMsg` zu groß, so werden CUT/CUIT terminiert.

CUCB_SocketTcpRcv

Die Funktion `CUCB_SocketTcpRcv()` wird von der COM aufgerufen, wenn die Nutzdaten vom Socket bereit liegen.

Nach dem Verlassen der Funktion `CUCB_SocketTcpRcv()` darf auf *pMsg nicht mehr zugegriffen werden.

Werden die Nutzdaten auch außerhalb der Funktion `CUCB_SocketTcpRcv()` benötigt, müssen die Nutzdaten aus *`pMsg` in einen dafür angelegten Bereich kopiert werden.

i Falls die TCP Verbindung asynchron getrennt wird (nach einem Fehler oder auf einen Request der anderen Seite), wird `CUCB_SocketTcpRcv()` mit dataLength = 0 aufgerufen.

Durch diesen Aufruf wird der CUT signalisiert, dass sie den Socket schließen muss, um die Kommunikation neu zu synchronisieren.

Funktionsprototyp:

```
void CUCB_SocketTcpRcv( dword socket,
                        void *pMsg,
                        udword dataLength)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
socket	Socket, über den die Nutzdaten empfangen wurden.
pMsg	Der Parameter `pMsg` zeigt auf den Beginn der Nutzdaten **ohne** Ethernet-/IP-/TCP-Header.
dataLength	Die Länge der Nutzdaten in Bytes

Tabelle 305: Parameter

CUL_SocketClose

Die Funktion `CUL_SocketClose()` schließt einen zuvor erzeugten Socket.

Funktionsprototyp:

dword CUL_SocketClose(dword socket)

Parameter:

Die Funktion hat den folgenden Parameter:

Parameter	Beschreibung
socket	zuvor erzeugter Socket

Tabelle 306: Parameter

Rückgabewert:

Es wird ein Error code (udword) zurückgegeben.

Die Error codes sind in der im Header-Datei cut.h definiert.

Error code	Beschreibung
CUL_OKAY	Socket geschlossen und eine Socket-Ressource wieder frei.
CUL_WRONG_SOCK	Socket nicht vorhanden

Tabelle 307: Rückgabewert

11.5.8 Timer-IF

CUL_GetTimeStampMS

Die Funktion `CUL_GetTimeStampMS()` liefert einen Millisekunden-Tick. Dieser ist geeignet in der CUT/CUIT eigene Timer zu implementieren. Der Zähler wird vom Quarz des COM-Prozessors abgeleitet und hat damit dieselbe Genauigkeit.

Funktionsprototyp:

udword CUL_GetTimeStampMS(void)

CUL_GetDateAndTime

Die Funktion `CUL_GetDateAndTime()` liefert an die übergebene Speicherstelle `*pSec` die Sekunden seit 1. Januar 1970, 00:00 und in `*pMsec` die zugehörigen Millisekunden. Die Werte werden mit der sicheren CPU abgeglichen und können je nach Parametrierung über SNTP (siehe Kapitel 9) extern synchronisiert werden.

Die Werte von `CUL_GetDateAndTime()` sollten **nicht** für Zeitmessungen, Timer oder ähnliches verwendet werden, da sie durch die Synchronisation und/oder durch den Anwender im Betrieb gestellt werden können.

Funktionsprototyp:

void CUL_GetDateAndTime(udword *pSec, udword *pMsec)

Restriktionen:

Ist der Speicher von pSec oder pMsec nicht im CUT Daten-Segment, werden CUT/CUIT terminiert.

11.5.9 Diagnose

Die Funktion `CUL_DiagEntry()` trägt ein Ereignis in die COM-Kurzzeitdiagnose ein, welches über das PADT ausgelesen werden kann.

Funktionsprototyp:

```
void CUL_DiagEntry(    udword severity,
                       udword code,
                       udword param1,
                       udword param2)
```

Parameter:

Die Funktion hat die folgenden Parameter:

Parameter	Beschreibung
severity	severity dient der Klassifizierung des Ereignisses 0x45 ('E') == Fehler, 0x57 ('W') == Warnung, 0x49 ('I') == Information
code	Der Anwender definiert den Parameter code mit einer beliebigen Nummer für entsprechende Ereignisse. Beim Eintritt des Ereignisses wird die Nummer in der Diagnose angezeigt.
param1, param2	Zusätzliche Informationen über das Ereignis

Tabelle 308: Parameter

11.6 Installation der Entwicklungsumgebung

In diesem Kapitel wird die Installation der Entwicklungsumgebung und die Erstellung einer ComUserTask beschrieben.

Die Entwicklungsumgebung befindet sich auf der Installations-CD, siehe Kapitel 11.2.

11.6.1 Installation der Cygwin-Umgebung

Die Cygwin-Umgebung ist erforderlich, da die GNU C Compiler Tools nur unter der Cygwin-Umgebung lauffähig sind.

Die Cygwin-Umgebung muss unter Windows 2000/XP/Vista installiert werden.

i Beachten Sie die Voraussetzungen zur Installation im Kapitel 11.2. **Deaktivieren Sie den Virenscanner** auf dem PC, auf dem Cygwin installiert werden soll, um Probleme bei der Installation von Cygwin zu verhindern.

Führen Sie die folgenden Schritte aus, um die Cygwin-Umgebung zu installieren:

Starten Sie das Setupprogramm zur Installation von Cygwin:

1. Kopieren Sie das Cygwin-Installationsarchiv cygwin_2005-03-11 von der Installations-CD auf Ihre lokale Festplatte (z. B. `Laufwerk C:\`).
2. Öffnen Sie im Windows-Explorer das Cygwin-Verzeichnis `C:\ cygwin_2005-03-11`.
3. Starten Sie die Installation von Cygwin mit einem Doppelklick auf die Datei **setup.exe**.
4. Klicken Sie im Cygwin-Dialogfenster auf die Schaltfläche **Weiter**, um das Setup auszuführen.

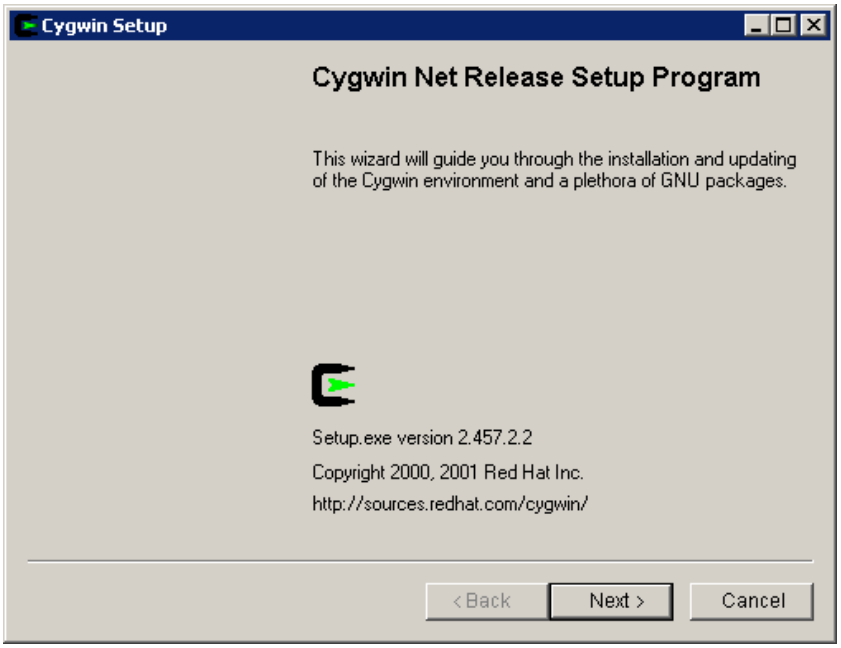

Cygwin Setup
Cygwin Net Release Setup Program
This wizard will guide you through the installation and updating of the Cygwin environment and a plethora of GNU packages.
Setup.exe version 2.457.2.2
Copyright 2000, 2001 Red Hat Inc.
http://sources.redhat.com/cygwin/
< Back
Next >
Cancel

Bild 83: Cygwin Setup-Dialog *Cygwin Setup*

Der Dialog Disable Virus Scanner erscheint, wenn der Virusscanner nicht deaktiviert wurde.

Führen Sie diesen Schritt aus, um den Virusscanner für die Installation von Cygwin zu deaktivieren.

i Deaktivieren Sie den Virusscanner vor der Cygwin-Installation, da es abhängig vom verwendeten Virusscanner vorkommen kann, dass dieses Fenster nicht erscheint, obwohl der Virusscanner läuft.

1. Wählen Sie **Disable Virus scanner**, um Probleme während der Installation durch den Virusscanner zu verhindern.
2. Klicken Sie auf die Schaltfläche **Weiter**, um die Eingabe zu bestätigen.

Wählen Sie im Dialog *Choose Installation Type* die Installations-Quelle von Cygwin aus:

1. Wählen Sie als Installations-Quelle **Install from Local Directory** aus.
2. Klicken Sie auf die Schaltfläche **Weiter**, um die Eingabe zu bestätigen.

Wählen Sie im Dialog *Choose Installation Directory* das Installations-Ziel von Cygwin aus:

1. Geben Sie das Verzeichnis an, in welches Cygwin installiert werden soll.
2. Übernehmen Sie alle weiteren Voreinstellungen des Dialogs.
3. Klicken Sie auf die Schaltfläche **Weiter**, um die Eingabe zu bestätigen.

Wählen Sie im Dialog *Select Local Package Directory* das Cygwin-Installationsarchiv

1. Geben Sie im Feld *Local Package Directory* das Cygwin-Installationsarchiv an, in dem sich die Installationsdateien befinden.
2. Klicken Sie auf die Schaltfläche **Weiter**, um die Eingabe zu bestätigen.

Wählen Sie Dialog *Select Packages* alle Packages zur Installation aus:

1. Wählen Sie den radio button **Curr**.
2. Klicken Sie im Anzeigefeld mehrmals langsam auf die Installationsoption neben **All**, bis **Install** für eine vollständige Installation aller packages angezeigt wird (ca. 1,86 GB Speicherbedarf).

i Achten Sie darauf, dass hinter jedem package Install steht.
Wenn die packages nicht vollständig Installiert werden, dann fehlen wichtige Funktionen, um später den C-Code der CUT zu compilieren!

3. Klicken Sie auf die Schaltfläche **Weiter**, um die Eingabe zu bestätigen.

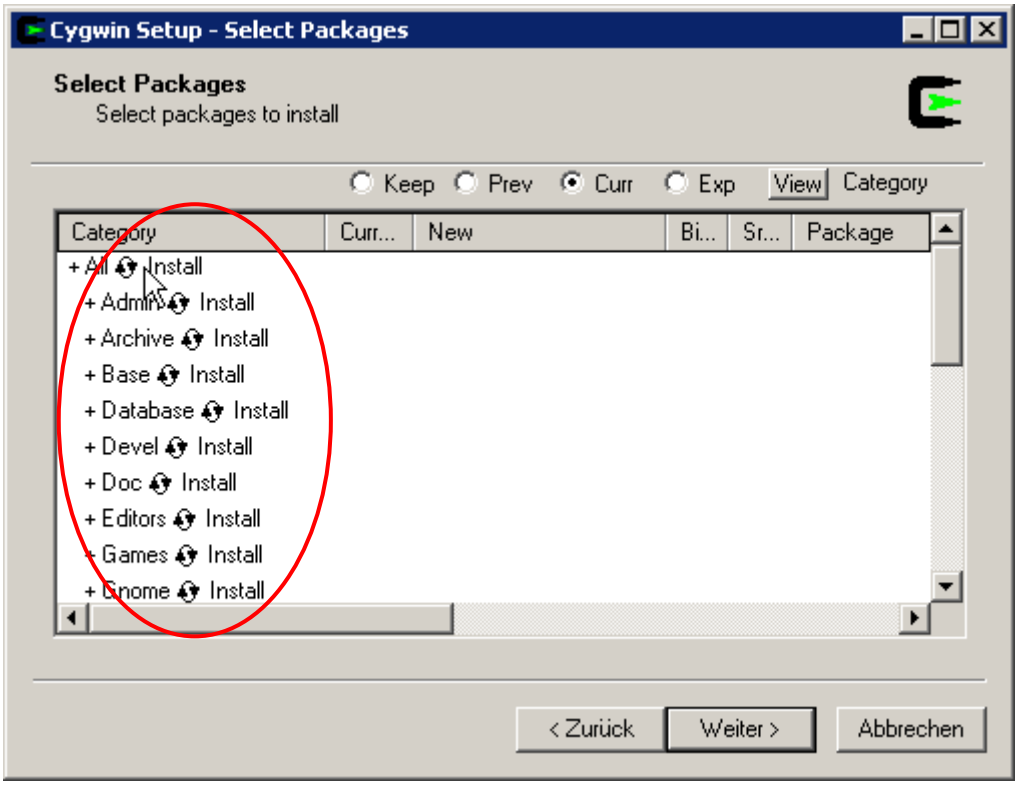


Bild 84: Cygwin Setup-Dialog *Select Packages*

Schließen Sie die Cygwin-Installation mit den folgenden Einträgen ab:

1. Wählen Sie Eintrag in das **Startmenü**.
2. Wählen Sie den Eintrag **Desktop-Icon**.
3. Klicken Sie auf die Schaltfläche **Fertigstellen**, um die Installation von Cygwin abzuschließen.

Cygwin Befehle	Beschreibung
cd (Name Verzeichnis)	Verzeichnis wechseln
cd ..	In höheres Verzeichnis wechseln
ls -l	Alle Dateien eines Verzeichnisses anzeigen
help	Übersicht über Bash Shell Kommandos

Tabelle 309: Befehle in Cygwin (Bash Shell)

11.6.2 Installation des GNU Compilers

Führen Sie die folgenden Schritte aus, um den GNU Compiler zu installieren:

1. Öffnen Sie im Windows-Explorer das Verzeichnis der Installations CD.
2. Doppelklicken Sie auf das zip-File `gcc-ppc-v3.3.2_binutils-v2.15.zip`.
3. Extrahieren Sie alle Dateien in das Cygwin-Verzeichnis (z.B. `C:\cygwin\…`). Der GNU Compiler wird im Cygwin-Verzeichnis in den Ordner **gcc-ppc** entpackt.

4. Tragen Sie in der Systemsteuerung die Umgebungsvariablen ein:
   - Öffnen Sie die Systemeigenschaften über das Windows-Startmenü **Einstellungen->Systemsteuerung->System**.
   - Wählen Sie das Register **Erweitert**.
   - Klicken Sie auf die Schaltfläche **Umgebungsvariablen**.
   - Wählen Sie im Feld *Systemvariablen* die Systemvariable **Path** und erweitern Sie die Systemvariable mit dem Eintrag: `C:\cygwin\gcc-ppc\bin`.

Kopieren Sie von der Installations-CD den Ordner cut_src in das Home-Verzeichnis.
Der Ordner cut_src enthält alle für die Erstellung eines ComUserTask benötigten "include"- und "lib"-Verzeichnisse. Die mitgelieferte Sourcedatei cut_cbf.c befindet sich standardmäßig im Verzeichnis ...\cut_src\cutapp\.

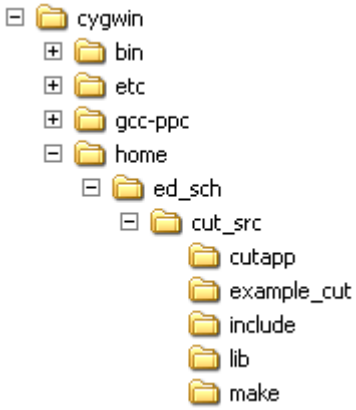


Bild 85: Strukturbaum von cygwin

---

i Wenn das Home-Verzeichnis nicht automatisch angelegt wurde, dann legen Sie das Home-Verzeichnis mit dem Windows-Explorer an (z.B. `C:\cygwin\home\User1`).

Wenn Sie ein anderes Home-Verzeichnis für die Cygwin Bash Shell anlegen wollen, dann müssen Sie die Batchdatei cygwin.bat mit dem Befehl `set Home` ergänzen.

```
@echo off

C:

chdir C:\cygwin\bin

set Home=C:\User1

bash --login -i
```

Bild 86: Batchdatei *Cygwin.bat*

11.7 Neues CUT-Projekt anlegen

Dieses Kapitel zeigt, wie ein neues CUT-Projekt angelegt wird und welche Dateien angepasst werden müssen.

---

i Das CUT-Projekt **example_cut** befindet sich fertig angepasst auf der CUT-CD.

Beim Anlegen weiterer neuer CUT-Projekte wird empfohlen, für jedes CUT-Projekt ein neues Verzeichnis unterhalb *...\cut_src* einzurichten.

Beispiel:

Zum Test wird das Verzeichnis *example_cut* angelegt, die C-Source heißt **example_cut.c**, die erzeugte ldb-Datei im Verzeichnis *make* heißt dann **example_cut.ldb**.

Erstellen Sie für eine neue ComUserTask den Ordner *example_cut.*

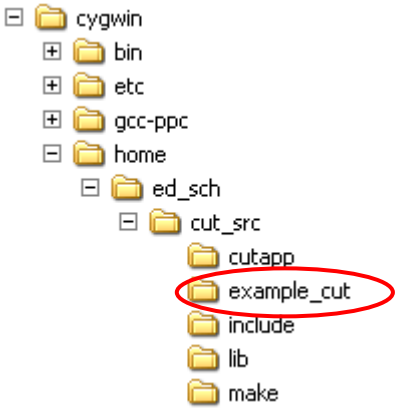


Bild 87: Strukturbaum von cygwin

1. Kopieren Sie die Dateien
   - cutapp.c
   - cutapp.mke
   - makefile

   aus dem Verzeichnis *cutapp* in das Verzeichnis *example_cut*.
2. Die Dateinamen müssen in den Loadable-Namen umbenannt werden (z.B. von **cutapp** in **example_cut**).

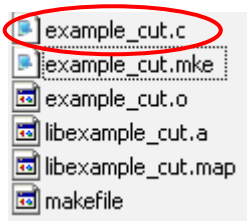


Bild 88: C-Code Datei im Ordner example_cut

Die Änderungen in der mke-Datei und im makefile müssen, wie in den nächsten Kapiteln beschrieben, bei einem neuen Projekt durchgeführt werden.

11.7.1 CUT-Makefiles

Konfiguration der CUT-Makefiles für unterschiedliche Sourcefiles und ldb-Dateien

Insgesamt müssen drei Makefiles, wie in den folgenden Absätzen beschrieben, angepasst werden.

Makefile mit der Erweiterung „.mke“

Die mke-Datei finden Sie im jeweiligen Quellcode-Verzeichnis
z.B. *cut_src\example_cut\example_cut.mke.*

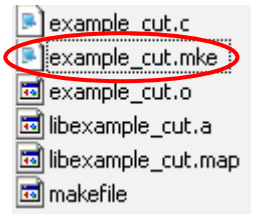


Bild 89: mke-Datei im Ordner example_cut

Führen Sie in der mke-Datei die folgenden Änderungen durch:

1. Der Variablen **module** muss der gleiche Loadable-Namen zugewiesen werden wie der .mke Datei (z.B. example_cut).
2. Der Variablen **c_sources** können eine oder mehrere C-Dateien zugewiesen werden, welche für die Erstellung des Ziel-Codes (Loadable-Datei) benötigt werden.

```
#############################################################################
#
# make file (DOS/NT)
# $Id: cutapp.mke 58869 2005-10-11 12:35:46Z es_fp $
#############################################################################
#
# assign name of module here (e.g. nl for NetworkLayer)
module= example_cut
#
# assign module sources here
sources=

c_sources= $(module).c
asm_sources=
```

Bild 90: mke-Datei ab der Zeile 1

Makefile

Die makefile-Datei finden Sie im jeweiligen Quellcode-Verzeichnis.
z.B. `cut_src\example_cut\makefile`

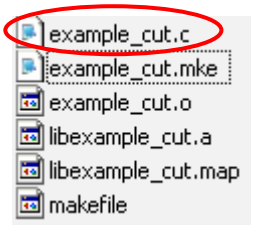


Bild 91: makefile im Ordner *example_cut*

Führen Sie in der makefile-Datei die folgenden Änderungen durch:

1. Ziehen Sie die Include-Zeile für die mke-Datei nach oben und ändern Sie die .mke Datei auf den aktuellen Namen.
2. Erweitern Sie den Make-Aufruf mit den beiden Variablen **SUBMOD_DIRS** und **CUT_NAME**.

```
all_objects:

include example_cut.mke

cut:
$(MAKE) -C ../make elf SUBMOD_DIRS=cut_src/$(module) CUT_NAME=$(module)


all_objects: $(c_objects) $(asm_objects) $(objects)
```

Bild 92: makefile ab der Zeile 34

Makefile mit der Erweiterung „makeinc.inc.app“

Als einmalige Änderung für dieses und alle weiteren CUT-Projekte wird der Name des CUT-Loadable über eine Make-Variable änderbar gemacht.

Die makeinc.inc.app-Datei finden Sie im cut_src-Verzeichnis
z.B. *cut_src\makeinc.inc.app.*

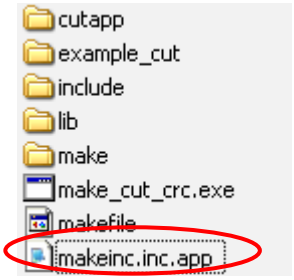


Bild 93: makeinc.inc.app Datei im Ordner *example_cut*

Führen Sie in der makeinc.inc.app-Datei die folgenden Änderungen durch:

1. Erweitern Sie die Datei mit der Variablen **CUT_NAME**.

```
all : lib$(module).$(LIBEXT)
@echo 'did make for module ['lib$(module).$(LIBEXT)']'

lib$(module).$(LIBEXT) : $(objects) $(c_objects) $(asm_objects) $(libraries)

SUBMOD2_LIBS=$(foreach lib,$(SUBMOD_LIBS),../../$(lib))

CUT_NAME=cut

makeAllLibs:
$(MAKE) -C ../../cut_src cut_src


makeLoadable:
@echo; \
BGTYPE=" $(CUT_NAME)"; \
if [ ! -f $$BGTYPE.map ] ; then \
echo "Error: MAP-Datei $$BGTYPE.map existiert nicht"; \
exit 1; \
fi; \
OS_LENGTH=$$(gawk '/___OS_LENGTH/ {print substr($$1,3,8)}' $$BGTYPE.map); \
echo; \
$(OBJCOPY) --strip-all --strip-debug -O binary $$BGTYPE.elf $$BGTYPE.bin;\
echo; \
echo "Building C3-Loadable-Binary ..."; \
$(MCRC) $$BGTYPE.bin 0 $$OS_LENGTH $$OS_LENGTH $$BGTYPE.ldb; \
echo; \

$(CUT_NAME).elf:  makeAllLibs $(SUBMOD2_LIBS)

elf:
@echo; test -f section.dld && $(MAKE) $(CUT_NAME).elf && $(MAKE) makeLoadable \
|| { echo "ERROR: Wrong subdir. Please invoke elf target only from make/ subdirectory."
&& echo && false ; } ;

# end of file: makeinc.inc
```

Bild 94: makeinc.inc.app ab der Zeile 247

11.7.2 C-Quellcode bearbeiten

Führen Sie die folgenden Schritte aus, um die Quellcodedatei zu öffnen:

1. Öffnen Sie das Projektverzeichnis *cut_src**example_cut*, das Sie in den vorangegangenen Schritten erstellt und konfiguriert haben.
2. Öffnen Sie mit einem Editor (z.B. Notepad) die C-Quellcodedatei mit der Erweiterung **.c**.

Eingangs- und Ausgangsvariablen konfigurieren

Führen Sie die folgenden Schritte aus, um die Ein- und Ausgangsvariablen in der Quellcodedatei zu konfigurieren:

1. Die Datengröße der Variablen, die im Register **Daten Ausgänge** in SILworX angelegt werden sollen, muss in der Quellcodedatei im Array **CUT_PDI[X]** angelegt werden.
2. Die Datengröße der Variablen, die im Register **Daten Eingänge** in SILworX angelegt werden sollen, muss in der Quellcodedatei im Array **CUT_PDO[X]** angelegt werden.

Startfunktion im CUT

Die C-Funktion `void CUCB_TaskLoop (udword mode)` ist die Startfunktion und wird vom Anwenderprogramm zyklisch aufgerufen.

Beispielcode „example_cut.c“

Der folgende C-Code kopiert den Wert vom Eingang **CUT_PDI[0]** in den Ausgang **CUT_PDO[0]** und gibt den Wert unverändert an das SILworX Anwenderprogramm zurück.

Hinweis Der C-Code **example_cut.c** befindet sich auf der Installations-CD.

```
/* Example for the CUT implemetation */
#include "include/cut_types.h"
#include "include/cut.h"
#ifdef __cplusplus
extern "C" {
#endif
/*****************************************************************************/
/* SILworX Output Records (CPU->COM) */
uword CUT_PDI[1] __attribute__ ((section("CUT_PD_IN_SECT"), aligned(1)));
/* SILworX Input Records (COM->CPU) */
uword CUT_PDO[1] __attribute__ ((section("CUT_PD_OUT_SECT"), aligned(1)));
/*****************************************************************************/

/* Callback function for starting the CUT */
  void CUCB_TaskLoop(udword mode)
  {
    if (CUT_PDI[0] > CUT_PDO[0])/*This is executed only, if the          */
      {                         /*SILworX application program    */
                                /*was processed.                       */
                                /*The SILworX application program*/
                                /*adds the value 1 to CUT_PDO[0] and   */
                                /*writes the result into CUT_PDI[0]    */

       CUT_PDO[0] = CUT_PDI[0];/*Copies the value from input CUT_PDI[0] */
                               /*into output CUT_PDO[0] of the SPS     */
       if (CUT_PDO[0] == 65535)
          {CUT_PDO[0] = 0;}
      }
  }
/*****************************************************************************/
```

Bild 95: Bilduntertitel

```
/*****************************************************************************/
  void CUCB_AscRcvReady(udword comId)
  {
    CUL_DiagEntry(0x49, 1, comId, 0);
  }
/*****************************************************************************/
  void CUCB_AscSendReady(udword comId)
  {
    CUL_DiagEntry(0x49, 2, comId, 0);
  }
/*****************************************************************************/
  void CUCB_SocketTryAccept(dword serverSocket)
  {
    CUL_DiagEntry(0x49, 3, serverSocket, 0);
  }
/*****************************************************************************/
  void CUCB_SocketConnected(dword socket, bool Okay)
  {
    CUL_DiagEntry(0x49, 4, socket, Okay);
  }
/*****************************************************************************/
  void CUCB_SocketTcpRcv(dword socket, void *pMsg, udword dataLength)
  {
    CUL_DiagEntry(0x49, 5, socket, dataLength);
  }
/*****************************************************************************/
  void CUCB_SocketUdpRcv(dword socket, void *pMsg, udword packetLength,
                         udword dataLength)
  {
    CUL_DiagEntry(0x49, 6, socket, dataLength);
  }
/*****************************************************************************/
  void CUCB_IrqService(udword devNo)
  {
    CUL_DiagEntry(0x49, 7, devNo, 0);
  }
/*****************************************************************************/

#ifdef __cplusplus
} /* end extern "C" */
#endif

/* end of file */
```

Bild 96: C-Code example_cut.c

Erstellung des Ausführbaren Code (ldb-Datei)

Führen Sie die folgenden Schritte zur Erstellung des ausführbaren Codes (ldb-Datei) aus:

1. Starten Sie die **Cygwin Bash Shell.**
2. Wechseln Sie in das Verzeichnis */cut_src/example_cut/*..
3. Starten Sie die Codegenerierung mit der Eingabe `make cut`.
   Das Binärfile **cut.ldb** wird im Verzeichnis */cut_src/make/* automatisch erzeugt.
4. Wenn der CRC32 erzeugt wurde, dann wurde auch ausführbarer Code erzeugt (siehe rote Markierung in Bild 97).

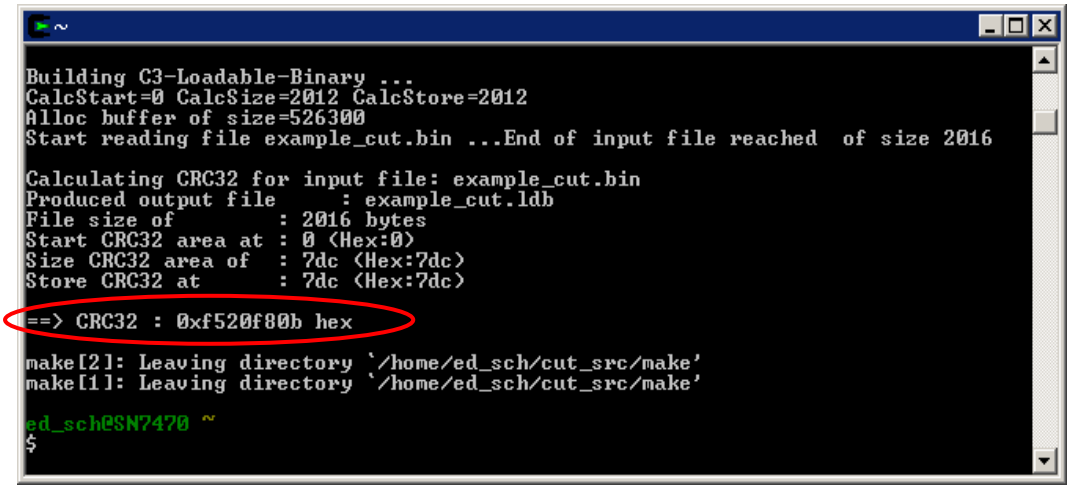

~
Building C3-Loadable-Binary ...
CalcStart=0 CalcSize=2012 CalcStore=2012
Alloc buffer of size=526300
Start reading file example_cut.bin ...End of input file reached of size 2016
Calculating CRC32 for input file: example_cut.bin
Produced output file : example_cut.ldb
File size of : 2016 bytes
Start CRC32 area at : 0 (Hex:0)
Size CRC32 area of : 7dc (Hex:7dc)
Store CRC32 at : 7dc (Hex:7dc)
==> CRC32 : 0xf520f80b hex
make[2]: Leaving directory `/home/ed_sch/cut_src/make'
make[1]: Leaving directory `/home/ed_sch/cut_src/make'
ed_sch@SN7470 ~
$

Bild 97: Cygwin Bash Shell

Dieser ausführbare Code (ldb-Datei) muss in das Projekt als ComUserTask geladen werden (siehe Kapitel 11.7.3).

11.7.3 ComUserTask in das Projekt einbinden

Führen Sie in SILworX die folgenden Schritte aus, um den ComUserTask in das Projekt einzubinden:

ComUserTask anlegen

So legen Sie eine neue ComUserTask an:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Im Kontextmenü von Protokolle **Neu**, **ComUserTask** wählen, um eine neue ComUserTask hinzuzufügen.
3. Im Kontextmenü der ComUserTask **Eigenschaften** das **COM-Modul** auswählen. Standardeinstellungen können für die erste Konfiguration beibehalten werden.

i Es kann immer nur ein ComUserTask pro Ressource angelegt werden.

Programmcode in das Projekt laden

So laden Sie eine neue ComUserTask in das Projekt:

1. Im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle** öffnen.
2. Rechtsklick auf **ComUserTask** und im Kontextmenü **User Task laden** wählen. Öffnen Sie das Verzeichnis *.../cut_src/make/*.
3. Wählen Sie die **ldb-Datei** aus, die im ComUserTask ausgeführt werden soll.

i Durch erneutes Laden des ausführbaren Codes (ldb-Datei) können neue Versionen der ldb-Datei übernommen werden. Der Inhalt der ldb-Datei wird beim Laden nicht auf Korrektheit geprüft. Die ldb-Datei wird anschließend zusammen mit der Ressourcekonfiguration im Projekt compiliert und kann in die Steuerung geladen werden. Wird die ldb-Datei verändert muss das Projekt erneut kompiliert und geladen werden.

Variablen mit dem CUT verbinden

Der Anwender kann eine nicht sicherheitsgerichtete Prozessdatenkommunikation zwischen der sicheren CPU und der nicht sicheren COM (CUT) definieren. Dabei können in jeder Richtung maximal 16kByte Daten ausgetauscht werden.

Erstellen Sie die beiden folgenden Globalen Variablen:

Variable	Typ
COM_CPU	UINT
CPU_COM	UINT

Prozessvariablen verbinden

Prozessvariablen im ComUserTask:

1. Rechtsklick auf **ComUserTask** und im Kontextmenü **Edit** wählen.
2. Im Dialog **Edit** das Register **Prozessvariablen** wählen.

Ausgangsvariablen (CPU->COM)

In das Register **Ausgangsvariablen** werden die Variablen eingetragen, die von der CPU zur COM übertragen werden sollen.

Name	Typ	Offset	Globale Variable
CPU_COM	UINT	0	CPU_COM

Tabelle 310: Ausgangsvariablen (CPU->COM)

1. Aus der Objektauswahl die Globalen Variablen zum versenden per Drag&Drop in das Register **Ausgangsvariablen** ziehen.
2. Rechtsklick auf eine leere Stelle im Bereich **Ausgangsvariablen** um Kontextmenü zu öffnen.
3. Im Kontextmenü **Neue Offsets** wählen, um die Offsets der Variablen neu zu generieren.

Eingangsvariablen (COM->CPU)

In das Register **Eingangsvariablen** werden die Variablen eingetragen, die von der COM zur CPU übertragen werden sollen.

Name	Typ	Offset	Globale Variable
COM_CPU	UINT	0	COM_CPU

Tabelle 311: Eingangsvariablen (COM->CPU)

1. Aus der Objektauswahl die Globalen Variablen zum empfangen per Drag&Drop in das Register **Eingangsvariablen** ziehen.
2. Rechtsklick auf eine leere Stelle im Bereich **Eingangsvariablen** um Kontextmenü zu öffnen.
3. Im Kontextmenü **Neue Offsets** wählen, um die Offsets der Variablen neu zu generieren.

ComUserTask Konfiguration verifizieren:

1. Im Strukturbaum **Konfiguration, Ressource, Protokolle, ComUserTask** wählen.
2. Rechtsklick auf **Verifikation**, um die CUT Konfiguration zu verifizieren.
3. Einträge im Logbuch sorgfältig überprüfen, gegebenenfalls korrigieren.

Erstellen Sie das SILworX Anwenderprogramm

So erstellen Sie das SILworX Anwenderprogramm

1. Im Strukturbaum **Konfiguration, Ressource** und im Kontextmenü **Edit** wählen.
2. Aus der Objektauswahl die Globalen Variablen **COM_CPU** und **CPU_COM** per Drag & Drop aus der Objektauswahl in das Zeichenfeld ziehen.
3. Erstellen Sie das Anwenderprogramm wie in der folgenden Abbildung dargestellt.

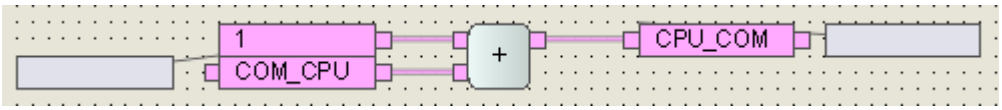


Bild 98: SILworX Programm-Editor

So Konfigurieren Sie das Schedule-Intervall [ms]

1. Rechtsklick auf **ComUserTask** und im Kontextmenü **Eigenschaften** wählen.
2. Im Eingabefeld *Schedule-Intervall [ms]* eintragen, in welchen Intervallen die ComUserTask aufgerufen werden soll.

i Die Konfiguration des ComUserTask muss mit dem Anwenderprogramm der Ressource neu compiliert und in die Steuerung übertragen werden, bevor dieser in der HIMax wirksam wird.

So Testen Sie die ComUserTask mit dem Online-Test

1. Im Strukturbaum **Konfiguration, Ressource, Programm** wählen.
2. Rechtsklick auf **Programm** und im Kontextmenü **Online** wählen. Führen Sie einen System-Login durch.

I

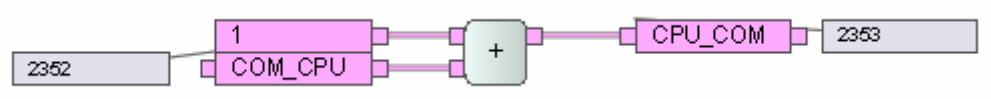


Bild 99: SILworX Online-Test

Funktion des SILworX Anwenderprogramm:

Das SILworX Anwenderprogramm addiert zu dem Signal **COM_CPU** (Daten-Eingänge) den Wert **1** und übergibt das Ergebnis dem Signal **CPU_COM** (Daten-Ausgänge).

Beim nächsten CUT-Aufruf (Schedule-Intervall [ms]) wird das Signal **CPU_COM** an die CUT-Funktion (siehe Beispielcode Kapitel 11.7.2) übergeben.

Die ComUserTask empfängt das Signal **CPU_COM** und sendet den Wert mit dem Signal **COM_CPU** unverändert zurück.

11.7.4 Fehler beim Laden einer Konfiguration mit CUT

Laufzeitprobleme (z.B. COM-User-Task in endlos-Schleife):

Ursache für Laufzeitprobleme:

Wenn in dem betreffenden CUT-Quellcode eine Schleife programmiert wurde, die sehr lange läuft kann dies zu einer „Verklemmung“ des COM-Prozessors führen.

Das hat zur Folge, dass zur Steuerung keine Verbindung mehr hergestellt werden kann und das Löschen der Ressourcenkonfiguration nicht mehr möglich ist.

Lösung: Reset des HIMax Kommunikationsmoduls oder der HIMatrix Steuerung:

- Führen Sie in der Online-Ansicht des Hardware Editors über die Funktion **Wartung/Service, Modul (Neustart)** einen Reset des Kommunikationsmoduls aus (Reset der HIMatrix über Reset-Taster, siehe Datenblatt der Steuerung).
- Erzeugen Sie eine neue CUT (möglichst ohne Laufzeitfehler, Endlosschleife).
- Laden Sie die CUT (ldb-Datei) in das Projekt.
- Generieren Sie den Code.
- Laden Sie die den Code in die Steuerung.

12 Allgemein

12.1 Konfiguration der Funktionsbausteine

Die Feldbus Protokolle und die zugehörigen Funktionsbausteine laufen auf dem COM-Modul der HIMax Steuerung. Daher müssen diese Funktionsbausteine im SILworX Strukturbaum unter **Konfiguration, Ressource, Protokolle...** angelegt werden.

Um diese Funktionsbausteine auf dem COM-Modul zu steuern, können im Anwenderprogramm von SILworX Funktionsbausteine angelegt werden (siehe Kapitel 12.1.1), die wie Standard-Funktionsbausteine verwendet werden können.

Die Verbindung der Funktionsbausteine im Anwenderprogramm von SILworX mit den entsprechenden Funktionsbausteinen im Strukturbaum von SILworX erfolgt über gemeinsame Variablen. Diese müssen zuvor vom Anwender im Variableneditor erstellt werden.

12.1.1 Beschaffung der Funktionsbausteinbibliotheken

Die Funktionsbausteinbibliotheken für PROFIBUS DP und TCP Send/Receive muss über die Funktion *Wiederherstellen...* (Kontextmenü des Projekts) dem Projekt hinzugefügt werden.

Die Funktionsbausteinbibliothek erhalten Sie auf Anfrage vom HIMA Support.

Tel.: +49-(0)6202-709 -185

-259

-261

E-Mail: support@hima.com

12.1.2 Konfiguration der Funktionsbausteine im Anwenderprogramm

Die benötigten Funktionsbausteine können per Drag&Drop in das Anwenderprogramm kopiert werden. Konfigurieren Sie die Eingänge und Ausgänge nach der Beschreibung des jeweiligen Funktionsbausteins.

Oberer Teil des Funktionsbausteins

Der obere Teil des Funktionsbausteins entspricht der Benutzerschnittstelle über die der Funktionsbaustein vom Anwenderprogramm gesteuert wird.

Hier werden die Variablen verbunden, die im Anwenderprogramm verwendet werden. Das Präfix "A" steht für "Applikation".

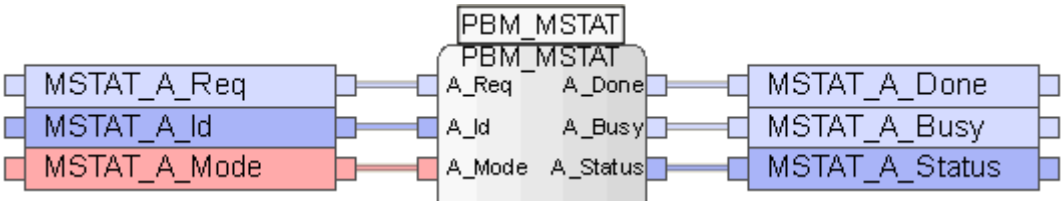


Bild 100: Funktionsbaustein PNM_MSTST (oberer Teil)

Unterer Teil des Funktionsbausteins

Der untere Teil des Funktionsbausteins stellt die Verbindung zum Funktionsbaustein (im Strukturbaum von SILworX) dar.

Hier werden die Variablen verbunden, die mit dem Funktionsbaustein im Strukturbaum von SILworX verbundenen werden müssen. Das Präfix "F" steht für "Field".

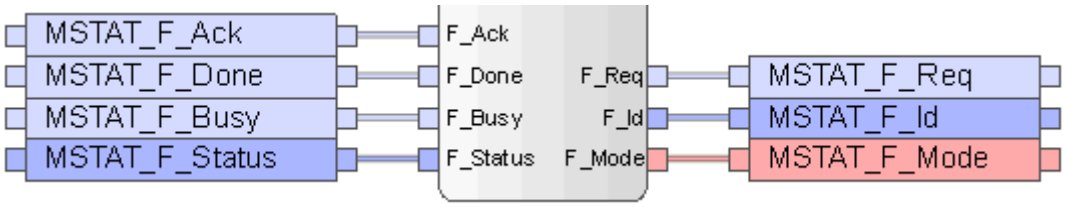


Bild 101: Funktionsbaustein PNM_MSTST (unterer Teil)

12.1.3 Konfiguration der Funktionsbausteine im Strukturbaum von SILworX

So konfigurieren Sie den Funktionsbaustein im Strukturbaum von SILworX:

1. Wählen Sie im Strukturbaum **Konfiguration**, **Ressource**, **Protokolle**, z. B. **PROFIBUS Master**.
2. Klicken Sie mit der rechten Maustaste auf **Funktionsbausteine** und wählen Sie **Neu**.
3. Wählen Sie den passenden Funktionsbaustein (im Strukturbaum von SILworX) aus.

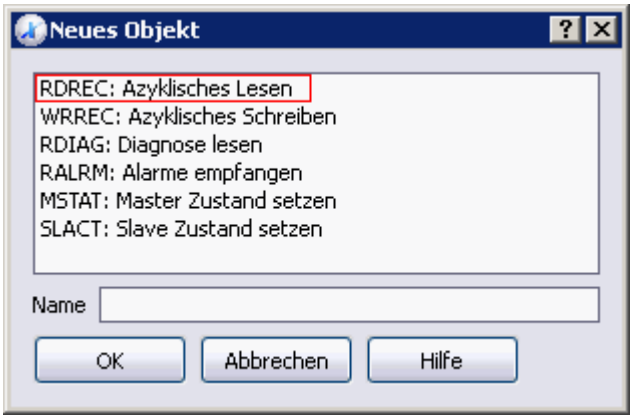


Bild 102: Auswahl Funktionsbausteine

Die Eingänge des Funktionsbausteins (Häkchen in Spalte Eingangsvariable) müssen mit den gleichen Variablen verbunden werden, die mit den *F-Ausgängen* des Funktionsbausteins im Anwenderprogramm verbunden sind.

Die Ausgänge des Funktionsbausteins (kein Häkchen in Spalte Eingangsvariable) müssen mit den gleichen Variablen verbunden werden, die mit den *F-Eingängen* des Funktionsbausteins im Anwenderprogramm verbunden sind.

Systemvariablen

F	Name	Datentyp	Transfer-Operation	Eingangsvariable	Globale Variable
1	ACK	BOOL	5	☑	MSTAT_F_Ack
2	BUSY	BOOL	5	☑	MSTAT_F_Busy
3	DONE	BOOL	5	☑	MSTAT_F_Done
4	REQ	BOOL	5	☐	MSTAT_F_Req
5	M_ID	DWORD	5	☐	MSTAT_F_Id
6	STATUS	DWORD	5	☑	MSTAT_F_Status
7	MODE	INT	5	☐	MSTAT_F_Mode

Bild 103: Systemvariablen des Funktionsbausteins MSTAT

12.2 Maximale Kommunikationszeitscheibe

Die maximale Kommunikationszeitscheibe ist die zugeteilte Zeit in Millisekunden (ms) pro CPU-Zyklus, innerhalb welcher das Prozessormodul die Kommunikationsaufgaben abarbeitet.
Können nicht alle in einem CPU-Zyklus anstehenden Kommunikationsaufgaben ausgeführt werden, erfolgt die komplette Übertragung der Kommunikationsdaten über mehrere CPU-Zyklen (Anzahl der Kommunikationszeitscheiben > 1).

Ermitteln der maximalen Kommunikationszeitscheibe

1. Betreiben Sie das HIMax System unter voller Last:
   Alle Kommunikationsprotokolle sind in Betrieb (safe**ethernet** und Standardprotokolle).
2. Öffnen Sie das **Control Panel** und wählen Sie im Strukturbaum das Verzeichnis **Statistiken**.
3. Öffnen Sie **Cyc.Async** um die Anzahl der Kommunikationszeitscheiben auszulesen.
4. Öffnen Sie **Zeit.Async** um die Dauer einer Kommunikationszeitscheibe auszulesen.

i Für die Berechnungen der zulässigen maximalen Reaktionszeiten, siehe Kapitel 4.7 gilt die Bedingung, daß die Anzahl der Kommunikationszeitscheiben = 1 ist.

Die Dauer der Kommunikationszeitscheibe ist so hoch einzustellen, daß der CPU-Zyklus die vom Prozess vorgegebene Watchdog-Zeit nicht überschreiten kann, wenn er die Kommunikationszeitscheibe ausnutzt.

Anhang

Glossar

Begriff	Beschreibung
ARP	Address Resolution Protocol: Netzwerkprotokoll zur Zuordnung von Netzwerkadressen zu Hardwareadressen
AI	Analog Input, Analoger Eingang
Connector Board	Anschlusskarte für HIMax Modul
COM	Kommunikationsmodul
CRC	Cyclic Redundancy Check, Prüfsumme
DI	Digital Input, digitaler Eingang
DO	Digital Output, digitaler Ausgang
EMV	Elektromagnetische Verträglichkeit
EN	Europäische Normen
ESD	ElectroStatic Discharge, elektrostatische Entladung
FB	Feldbus
FBS	Funktionsbausteinsprache
FTA	Field Termination Assembly
FTZ	Fehlertoleranzzeit
ICMP	Internet Control Message Protocol: Netzwerkprotokoll für Status- und Fehlermeldungen
IEC	Internationale Normen für die Elektrotechnik
MAC-Adresse	Hardware-Adresse eines Netzwerkanschlusses (Media Access Control)
PADT	Programming and Debugging Tool (nach IEC 61131-3), PC mit SILworX
PE	Schutzerde
PELV	Protective Extra Low Voltage: Funktionskleinspannung mit sicherer Trennung
PES	Programmierbares Elektronisches System
PFD	Probability of Failure on Demand: Wahrscheinlichkeit eines Fehlers bei Anforderung einer Sicherheitsfunktion
PFH	Probability of Failure per Hour: Wahrscheinlichkeit eines gefahrbringenden Ausfalls pro Stunde
R	Read
Rack-ID	Identifikation eines Basisträgers (Nummer)
rückwirkungsfrei	Es seien zwei Eingangsschaltungen an dieselbe Quelle (z.B. Transmitter) angeschlossen. Dann wird eine Eingangsschaltung „rückwirkungsfrei“ genannt, wenn sie die Signale der anderen Eingangsschaltung nicht verfälscht.
R/W	Read/Write
SB	Systembus (-modul)
SELV	Safety Extra Low Voltage: Schutzkleinspannung
SFF	Safe Failure Fraction, Anteil der sicher beherrschbaren Fehler
SIL	Safety Integrity Level (nach IEC 61508)
SILworX	Programmiersoftware für HIMax
SNTP	Simple Network Time Protocol (RFC 1769)
SRS	System.Rack.Slot
SW	Software
TMO	Timeout
W	Write
WD	Watchdog
WDZ	Watchdog-Zeit

Abbildungsverzeichnis

Bild 1: Konfigurationsdialog für CPU- und COM Modul in SILworX ... 24
Bild 2: Systemstrukturen ... 33
Bild 3: Aufbau zur Konfiguration einer redundante Verbindung ... 34
Bild 4: Strukturbaum der Ressource ... 34
Bild 5: Werte der Parameter einer redundanten safe**ethernet**-Verbindung ... 35
Bild 6: Detailansicht safe**ethernet** Editor ... 35
Bild 7: Redundante Verbindung zwischen zwei HIMax Steuerungen ... 43
Bild 8: Redundante Verbindung zweier HIMax Steuerungen über eine Leitung ... 43
Bild 9: safe**ethernet** Verbindung zweier HIMax Steuerungen ... 48
Bild 10: safe**ethernet** Verbindung einer HIMax mit einer HIMatrix-Steuerung ... 49
Bild 11: safe**ethernet** Verbindung mit in RIOs ... 49
Bild 12: safe**ethernet** Verbindung zweier HIMax mit einer HIMatrix Steuerung ... 50
Bild 13: safe**ethernet** Verbindung zwischen Ressource A1 im Projekt A und der Ressource B1 im Projekt B ... 55
Bild 14: Variante Projekt A als lokales Projekt ... 56
Bild 15: Variante Projekt B als lokales Projekt ... 56
Bild 16: Aufbau zur Kommunikation zwischen SILworX und ELOP II Factory ... 57
Bild 17: HIMatrix Proxy-Ressource ... 58
Bild 18: Parameter einer safe**ethernet**-Verbindung zur Proxy-Ressource ... 59
Bild 19: Safe**ethernet**-Verbindung exportieren ... 60
Bild 20: Verbindungen in ELOP II Factory importieren ... 61
Bild 21: Peer-to-Peer-Editor in ELOP II Factory ... 61
Bild 22: Sendesignale in ELOP II Factory zuweisen ... 62
Bild 23: Empfangssignale in ELOP II Factory zuweisen ... 62
Bild 24: Control Panel zur Verbindungsüberwachung ... 63
Bild 25: Strukturbaum des PROFINET-IO Controllers ... 70
Bild 26: Kommunikation über PROFINET IO ... 82
Bild 27: Kommunikation über PROFIBUS DP ... 94
Bild 28: HIMax PROFIBUS DP Slave mit Modulen ... 97
Bild 29: Benutzerdatenfeld ... 100
Bild 30: Dialogfenster Verifikation ... 100
Bild 31: Eigenschaften des PROFIBUS DP Master ... 101
Bild 32: Eigenschaften des PROFIBUS DP Slave ... 102
Bild 33: Isochroner PROFIBUS DP Zyklus ... 111
Bild 34: Dialog *User Parameter bearbeiten* ... 121
Bild 35: Funktionsbaustein MSTAT ... 123
Bild 36: Funktionsbaustein RALRM ... 126
Bild 37: Funktionsbaustein RDIAG ... 130
Bild 38: Funktionsbaustein RDREC ... 134
Bild 39: Funktionsbaustein SLACT ... 137
Bild 40: Funktionsbaustein WRREC ... 140
Bild 41: Hilfsfunktionsbaustein ACTIVE ... 143
Bild 42: Hilfsfunktionsbaustein ALARM ... 144
Bild 43: Hilfsfunktionsbaustein DEID ... 145
Bild 44: Hilfsfunktionsbaustein ID ... 146
Bild 45: Hilfsfunktionsbaustein NSLOT ... 147
Bild 46: Hilfsfunktionsbaustein SLOT ... 147
Bild 47: Hilfsfunktionsbaustein STDDIAG ... 148
Bild 48: Kommunikation über Modbus TCP ... 164
Bild 49: Modbus-Netzwerk ... 179
Bild 50: Modbus Gateway ... 182
Bild 51: Serieller Modbus ... 185
Bild 52: Modbus Telegramm ... 185
Bild 53: Verschaltung HIMax und Siemens Steuerung ... 211
Bild 54: Datenübertragung zwischen HIMax und Siemens Steuerung ... 212
Bild 55: Variablenliste im Siemens-Baustein UDT1 ... 213
Bild 56: Variablenliste im Siemens-Baustein DB1 ... 214
Bild 57: SIMATIC-Symboleditor ... 214
Bild 58: Funktionsplan zum Empfangen ... 215

Bild 59: Funktionsplan zum Senden ... 216
Bild 60: Eigenschaften der TCP-Verbindung in SILworX ... 217
Bild 61: Variablenliste Siemens ... 226
Bild 62: Variablenliste HIMax ... 226
Bild 63: Funktionsbaustein TCP_Reset ... 228
Bild 64: Funktionsbaustein TCP_Send ... 231
Bild 65: Funktionsbaustein TCP_Receive ... 234
Bild 66: Funktionsbaustein TCP_ReceiveLine ... 238
Bild 67: Funktionsbaustein TCP_ReceiveVar ... 242
Bild 68: Aufbau der Datenpakete ... 243
Bild 69: Redundanter X-OPC Betrieb ... 258
Bild 70: Installationsroutine des X-OPC-Servers ... 259
Bild 71: Installationsroutine des X-OPC-Servers ... 259
Bild 72: Mauelle Einstellung der ClassID des zweiten X-OPC-Servers ... 260
Bild 73: Einstellung für X-OPC-Server automatisch starten ... 261
Bild 74: Redundanter X-OPC Betrieb ... 262
Bild 75: Redundanter X-OPC Betrieb ... 263
Bild 76: Detailansicht der safe**ethernet**-Verbindung ... 264
Bild 77: Alarm & Event Editor ... 266
Bild 78: Redundanter X-OPC Betrieb ... 267
Bild 79: safe**ethernet** Editor ... 270
Bild 80: Detailansicht der safe**ethernet**-Verbindung ... 271
Bild 81: Fünf Bereiche eines skalaren Ereignisses ... 273
Bild 82: Prozessdatenaustausch zwischen CPU und COM (CUT) ... 284
Bild 83: Cygwin Setup-Dialog *Cygwin Setup* ... 313
Bild 84: Cygwin Setup-Dialog *Select Packages* ... 315
Bild 85: Strukturbaum von cygwin ... 316
Bild 87: Strukturbaum von cygwin ... 317
Bild 88: C-Code Datei im Ordner example_cut ... 317
Bild 89: mke-Datei im Ordner example_cut ... 318
Bild 90: mke-Datei ab der Zeile 1 ... 318
Bild 91: makefile im Ordner *example_cut* ... 319
Bild 92: makefile ab der Zeile 34 ... 319
Bild 93: makeinc.inc.app Datei im Ordner *example_cut* ... 319
Bild 94: makeinc.inc.app ab der Zeile 247 ... 320
Bild 95: Bilduntertitel ... 321
Bild 96: C-Code example_cut.c ... 322
Bild 97: Cygwin Bash Shell ... 323
Bild 98: SILworX Programm-Editor ... 326
Bild 99: SILworX Online-Test ... 326
Bild 100: Funktionsbaustein PNM_MSTST (oberer Teil) ... 328
Bild 101: Funktionsbaustein PNM_MSTST (unterer Teil) ... 329
Bild 102: Auswahl Funktionsbausteine ... 329
Bild 103: Systemvariablen des Funktionsbausteins MSTAT ... 330

Tabellenverzeichnis

Tabelle 1: Zusätzlich geltende Handbücher 11
Tabelle 2: Normen für EMV-, Klima- und Umweltanforderungen 14
Tabelle 3: Allgemeine Bedingungen 14
Tabelle 4: Klimatische Bedingungen 14
Tabelle 5: Mechanische Prüfungen 15
Tabelle 6: Prüfungen der Störfestigkeit 15
Tabelle 7: Prüfungen der Störfestigkeit 15
Tabelle 8: Prüfungen der Störaussendung 15
Tabelle 9: Nachprüfung der Eigenschaften der Gleichstromversorgung 16
Tabelle 10: Verfügbare Standardprotokolle 19
Tabelle 11: Protokolle auf einem Kommunikationsmodul 20
Tabelle 12: Protokolle auf einem Kommunikationsmodul 20
Tabelle 13: Teilenummern 21
Tabelle 14: Beispiele für Teilenummern des COM-Moduls 21
Tabelle 15: Verfügbare Protokolle der HIMax 22
Tabelle 16: Eigenschaften Ethernet Schnittstellen 23
Tabelle 17: Konfigurationsparameter 26
Tabelle 18: Routing Parameter 27
Tabelle 19: Ethernet-Switch-Parameter 27
Tabelle 20: Register VLAN 28
Tabelle 21: Feldbusschnittstellen 29
Tabelle 22: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für PROFIBUS DP 30
Tabelle 23: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für Modbus 30
Tabelle 24: Pin-Belegung der D-SUB Anschlüsse FB1 und FB2 für RS232 30
Tabelle 25: Parameter safe**ethernet**-Protokoll 38
Tabelle 26: Register Systemvariablen im safe**ethernet**-Editor 42
Tabelle 27: Verfügbare Ethernet-Schnittstellen 42
Tabelle 28: Kombinationen für safe**ethernet**-Verbindungen 43
Tabelle 29: Anzeigefeld der safe**ethernet**-Verbindung 64
Tabelle 30: Übersicht PROFINET-IO Funktionsbausteine 65
Tabelle 31: Systemanforderung und Ausstattung für PROFINET-IO Controller 66
Tabelle 32: Eigenschaften PROFINET-IO Controller 66
Tabelle 33: Allgemeine Eigenschaften des PROFINET-IO Controller 69
Tabelle 34: Register Parameter des PROFINET-IO Device 70
Tabelle 35: Register Parameter im Eigenschaftendialog des DAP Moduls 71
Tabelle 36: Register Parameter der I/O PROFINET-IO Module 71
Tabelle 37: Eigenschaftendialog von Submodul Input 72
Tabelle 38: Edit Dialog von Submodul Input 73
Tabelle 39: Eigenschaftendialog von Submodul Input 73
Tabelle 40: Edit Dialog von Submodul Output 74
Tabelle 41: Eigenschaftendialog von Submodul Input/Output 75
Tabelle 42: Edit Dialog von Submodul Input und Output 76
Tabelle 43: Eigenschaftendialog von Application Relation 77
Tabelle 44: Eigenschaftendialog von Alarm CR 78
Tabelle 45: Eigenschaftendialog von Default Input CR 79
Tabelle 46: Dialog Edit von Default Input CR 79
Tabelle 47: Eigenschaftendialog von Default Output CR 80
Tabelle 48: Systemanforderung und Ausstattung für PROFINET-IO Controller 81
Tabelle 49: Eigenschaften PROFINET-IO Controller 81
Tabelle 50: Variablen im Ausgangsmodul [01] Out 2 Bytes_1 83
Tabelle 51: Variablen Ausgangsmodul [02] Out 8 Bytes_2 84
Tabelle 52: Variablen im Ausgangsmodul [03] Out 1 Byte_3 84
Tabelle 53: Variablen im Eingangsmodul [04] In 2 Bytes_4 84
Tabelle 54: Variablen im Eingangsmodul [05] In 1 Byte_5 85
Tabelle 55: Variablen im Eingangsmodul [001] Input 2 Byte: Modul_1 87
Tabelle 56: Variablen Eingangsmodul [002] Input 8 Byte: Modul_2 87
Tabelle 57: Variablen im Eingangsmodul [003] Input 1 Byte: Modul_3 88
Tabelle 58: Variablen im Ausgangsmodul [004] Output 2 Byte: Modul_4 88
Tabelle 59: Variablen im Ausgangsmodul [005] Output 1 Bytes: Modul_5 88
Tabelle 60: Allgemeine Eigenschaften des PROFINET-IO Device 89

Tabelle 61: Allgemeine Eigenschaften des PROFINET-IO Device 90
Tabelle 62: Allgemeine Eigenschaften des PROFINET-IO Device 91
Tabelle 63: Benötigte Ausstattung und Systemanforderungen 93
Tabelle 64: Eigenschaften PROFIBUS DP Master 93
Tabelle 65: Ausgänge HIMA PROFIBUS DP Slave 95
Tabelle 66: Eingänge HIMA PROFIBUS DP Slave 95
Tabelle 67: Variablen Eingangsmodul [000] DP-Input/ELOP-Export: 2 Bytes 98
Tabelle 68: Variablen Eingangsmodul [001] DP-Input/ELOP-Export: 8 Bytes 98
Tabelle 69: Variablen Eingangsmodul [002] DP-Input/ELOP-Export: 1 Byte 99
Tabelle 70: Variablen Ausgangsmodul [003] DP-Output/ELOP-Import: 2 Bytes 99
Tabelle 71: Variablen Ausgangsmodul [004] DP-Output/ELOP-Import: 1 Byte 99
Tabelle 72: Systemvariablen des PROFIBUS DP Master 103
Tabelle 73: Allgemeine Eigenschaften des PROFIBUS DP Master 104
Tabelle 74: Register Zeiten im Eigenschaftendialog des PROFIBUS DP Master 106
Tabelle 75: Register CPU/COM im Eigenschaftendialog des PROFIBUS DP Master 106
Tabelle 76: Sonstige Eigenschaften des PROFIBUS DP Master 107
Tabelle 77: HIMax Richtwerte für Token-Umlaufzeit bei verschiedenen Übertragungsraten 108
Tabelle 78: Übertragungszeit für ein Zeichen bei verschiedenen Übertragungsraten 109
Tabelle 79: Elemente zur Berechnung der Token-Umlaufzeit 109
Tabelle 80: Systemvariablen des PROFIBUS DP Slave 113
Tabelle 81: Register Parameter des PROFIBUS DP Slave 114
Tabelle 82: Register Gruppen im Eigenschaftendialog des PROFIBUS DP Slave 115
Tabelle 83: Register DP-V1 im Eigenschaftendialog des PROFIBUS DP Slave 115
Tabelle 84: Register Alarme im Eigenschaftendialog des PROFIBUS DP Slave 116
Tabelle 85: Register Daten im Eigenschaftendialog des PROFIBUS DP Slave 116
Tabelle 86: Register Modell im Eigenschaftendialog des PROFIBUS DP Slave 117
Tabelle 87: Register Features im Eigenschaftendialog des PROFIBUS DP Slave 117
Tabelle 88: Register Übertragungsraten im Eigenschaftendialog des PROFIBUS DP Slave 118
Tabelle 89: Register Azyklisch im Eigenschaftendialog des PROFIBUS DP Slave 118
Tabelle 90: GSD-Datei des HIMax PROFIBUS DP Slave 119
Tabelle 91: Beispiel: Blöcke 1 ... 4 des Benutzerdatenfeldes 121
Tabelle 92: Beispiel: Blöcke 1 ... 4 des Benutzerdatenfeldes 121
Tabelle 93: Übersicht PROFIBUS DP Funktionsbausteine 122
Tabelle 94: A-Eingänge Funktionsbaustein MSTAT 123
Tabelle 95: A-Ausgänge Funktionsbaustein MSTAT 123
Tabelle 96: F-Eingänge Funktionsbaustein MSTAT 124
Tabelle 97: F-Ausgänge Funktionsbaustein MSTAT 124
Tabelle 98: Eingangs-Systemvariablen 124
Tabelle 99: Ausgangs-Systemvariablen 125
Tabelle 100: A-Eingänge Funktionsbaustein RALRM 126
Tabelle 101: A-Ausgänge Funktionsbaustein RALRM 127
Tabelle 102: F-Eingänge Funktionsbaustein RALRM 127
Tabelle 103: F-Ausgänge Funktionsbaustein RALRM 127
Tabelle 104: Eingangs-Systemvariablen 128
Tabelle 105: Ausgangs-Systemvariablen 128
Tabelle 106: Alarmdaten 129
Tabelle 107: A-Eingänge Funktionsbaustein RALRM 130
Tabelle 108: A-Ausgänge Funktionsbaustein RALRM 130
Tabelle 109: F-Eingänge Funktionsbaustein RDIAG 131
Tabelle 110: F-Ausgänge Funktionsbaustein RDIAG 131
Tabelle 111: Eingangs-Systemvariablen 131
Tabelle 112: Ausgangs-Systemvariablen 132
Tabelle 113: Diagnosedaten 132
Tabelle 114: A-Eingänge Funktionsbaustein RDREC 134
Tabelle 115: A-Ausgänge Funktionsbaustein RDREC 134
Tabelle 116: F-Eingänge Funktionsbaustein RDREC 135
Tabelle 117: F-Ausgänge Funktionsbaustein RDREC 135
Tabelle 118: Eingangs-Systemvariablen 135
Tabelle 119: Ausgangs-Systemvariablen 136
Tabelle 120: Daten 136
Tabelle 121: A-Eingänge Funktionsbaustein SLACT 137
Tabelle 122: A-Ausgänge Funktionsbaustein SLACT 137

Tabelle 123: F-Eingänge Funktionsbaustein SLACT 138
Tabelle 124: F-Ausgänge Funktionsbaustein SLACT 138
Tabelle 125: Eingangs-Systemvariablen 138
Tabelle 126: Ausgangs-Systemvariablen 139
Tabelle 127: A-Eingänge Funktionsbaustein WRREC 140
Tabelle 128: A-Ausgänge Funktionsbaustein WRREC 140
Tabelle 129: F-Eingänge Funktionsbaustein WRREC 141
Tabelle 130: F-Ausgänge Funktionsbaustein WRREC 141
Tabelle 131: Eingangs-Systemvariablen 141
Tabelle 132: Ausgangs-Systemvariablen 142
Tabelle 133: Daten 142
Tabelle 134: Übersicht Hilfsfunktionsbausteine 143
Tabelle 135: Eingänge Hilfsfunktionsbaustein ACTIVE 143
Tabelle 136: Ausgänge Hilfsfunktionsbaustein ACTIVE 143
Tabelle 137: Eingänge Hilfsfunktionsbaustein ALARM 144
Tabelle 138: Ausgänge Hilfsfunktionsbaustein ALARM 145
Tabelle 139: Eingänge Hilfsfunktionsbaustein DEID 145
Tabelle 140: Ausgänge Hilfsfunktionsbaustein DEID 145
Tabelle 141: Eingänge Hilfsfunktionsbaustein ID 146
Tabelle 142: Ausgänge Hilfsfunktionsbaustein ID 146
Tabelle 143: Eingänge Hilfsbaustein NSLOT 147
Tabelle 144: Ausgänge Hilfsbaustein NSLOT 147
Tabelle 145: Eingänge Hilfsbaustein SLOT 148
Tabelle 146: Ausgänge Hilfsbaustein SLOT 148
Tabelle 147: Eingänge Hilfsbaustein STDDIAG 149
Tabelle 148: Ausgänge Hilfsbaustein STDDIAG 149
Tabelle 149: Fehlercodes Funktionsbausteine 150
Tabelle 150: Anzeigenfeld PROFIBUS Master 153
Tabelle 151: PROFIBUS DP Master Zustand 153
Tabelle 152: PROFIBUS DP Master Verhalten 153
Tabelle 153: LED FBx (PROFIBUS DP Master) 154
Tabelle 154: Ausstattung und Systemanforderungen HIMA PROFIBUS DP Slave 155
Tabelle 155: Eigenschaften HIMA PROFIBUS DP Slave 155
Tabelle 156: Systemvariablen PROFIBUS DP Slave 157
Tabelle 157: Slave Eigenschaften: Register Allgemein 159
Tabelle 158: Anzeigenfeld (PROFIBUS DP Slave) 160
Tabelle 159: LED FBx (PROFIBUS DP Slave) 161
Tabelle 160: Ausstattung und Systemanforderung Modbus Master 163
Tabelle 161: Eigenschaften Modbus Master Nach der Norm sind insgesamt drei Repeater zulässig, sodass maximal 121 Slaves pro serielle Schnittstelle eines Masters möglich sind. 163
Tabelle 162: Systemvariablen Modbus Master 169
Tabelle 163: Allgemeine Eigenschaften Modbus Master 170
Tabelle 164: Parameter COM/CPU 171
Tabelle 165: Modbus-Funktionscodes 172
Tabelle 166: Anforderungstelegramm Read Coils 175
Tabelle 167: Anforderungstelegramm Read Discrete Inputs 175
Tabelle 168: Anforderungstelegramm Read Holding Registers 175
Tabelle 169: Anforderungstelegramm Read Input Registers 176
Tabelle 170: Register Read Write Holding 177
Tabelle 171: Anforderungstelegramm Write Multiple Coils 177
Tabelle 172: Anforderungstelegramm Write Multiple Registers 178
Tabelle 173: Anforderungstelegramm Write Single Coil (05) 178
Tabelle 174: Anforderungstelegramm Write Single Register 178
Tabelle 175: Systemvariablen TCP/UDP-Slaves 180
Tabelle 176: Konfigurationsparameter 181
Tabelle 177: Verbindungsparameter Modbus Gateway 184
Tabelle 178: Statusvariablen Gateway-Slave 184
Tabelle 179: Verbindungsparameter Gateway-Slave 184
Tabelle 180: Parameter serieller Modbus Master 186
Tabelle 181: Systemvariablen Modbus Slave 186
Tabelle 182: Verbindungsparameter Modbus Master 187
Tabelle 183: Anzeigefeld Modbus Master 188

Tabelle 184: LED FBx beim Modbus Master 189
Tabelle 185: Ausstattung und Systemanforderung HIMA Modbus Slave 190
Tabelle 186: Eigenschaften Modbus Slave 190
Tabelle 187: Zulässige Steckplätze der redundanten Modbus Slave COM-Module 192
Tabelle 188: Register Eigenschaften Modbus Slave Set 194
Tabelle 189: Anzeigefeld Modbus Master 195
Tabelle 190: Register TCP- und UDP-Ports für HIMA Modbus Slave 196
Tabelle 191: Register Systemvariablen für HIMA Modbus Slave 197
Tabelle 192: Modbus-Funktionscodes des HIMA Modbus Slave 198
Tabelle 193: Register Variablen im Register Bereich des Modbus Slave 202
Tabelle 194: Bit Variablen im Bit Bereich des Modbus Slave 203
Tabelle 195: Lasche Offsets für HIMA Modbus Slave 204
Tabelle 196: Aus dem Register Bereich in den Bit Bereich gespiegelte Variablen 205
Tabelle 197: Aus dem Bit Bereich in den Register Bereich gespiegelte Variablen 206
Tabelle 198: Anzeigefeld Modbus Slave 208
Tabelle 199: Anzeigefeld Masterdaten 208
Tabelle 200: LED FBx beim Modbus Slave 209
Tabelle 201: Fehlercodes Modbus TCP/IP 209
Tabelle 202: Systemanforderung und Ausstattung S&R TCP 210
Tabelle 203: Eigenschaften S&R TCP 210
Tabelle 204: Konfiguration HIMax Steuerung 212
Tabelle 205: Konfiguration Siemens SIMATIC 300 212
Tabelle 206: Globale Variablen 217
Tabelle 207: Variablen für Empfangsdaten 218
Tabelle 208: Variablen für Sendedaten 218
Tabelle 209: Systemvariablen S&R TCP 219
Tabelle 210: Allgemeine Eigenschaften S&R TCP 219
Tabelle 211: Parameter COM/CPU 220
Tabelle 212: Systemvariablen 221
Tabelle 213: Eigenschaften S&R TCP-Verbindung 223
Tabelle 214: Funktionsbausteine für S&R TCP Verbindungen 227
Tabelle 215: A-Eingänge Funkionsbaustein TCP_Reset 228
Tabelle 216: A-Ausgänge Funktionsbaustein TCP_Reset 228
Tabelle 217: F-Eingänge Funktionsbaustein TCP_Reset 229
Tabelle 218: F-Ausgänge Funktionsbaustein TCP_Reset 229
Tabelle 219: Eingangs-Systemvariablen 229
Tabelle 220: Ausgangs-Systemvariablen 230
Tabelle 221: A-Eingänge Funktionsbaustein TCP_Send 231
Tabelle 222: A-Ausgänge Funktionsbaustein TCP_Send 231
Tabelle 223: F-Eingänge Funktionsbaustein TCP_Send 232
Tabelle 224: F-Ausgänge Funktionsbaustein TCP_Send 232
Tabelle 225: Eingangs-Systemvariablen 232
Tabelle 226: Ausgangs-Systemvariablen 233
Tabelle 227: Sende-Daten 233
Tabelle 228: A-Eingänge Funktionsbaustein TCP_Receive 234
Tabelle 229: A-Ausgänge Funktionsbaustein TCP_Receive 235
Tabelle 230: A-Eingänge Funktionsbaustein TCP_Receive 235
Tabelle 231: F-Ausgänge Funktionsbaustein TCP_Receive 235
Tabelle 232: Eingangs-Systemvariablen 236
Tabelle 233: Ausgangs-Systemvariablen 236
Tabelle 234: Empfangsvariablen 236
Tabelle 235: A-Eingänge Funktionsbaustein TCP_ReceiveLine 238
Tabelle 236: A-Ausgänge Funktionsbaustein TCP_ReceiveLine 239
Tabelle 237: F-Eingänge Funktionsbaustein TCP_ReceiveLine 239
Tabelle 238: F-Ausgänge Funktionsbaustein TCP_ReceiveLine 239
Tabelle 239: Eingangs-Systemvariablen 240
Tabelle 240: Ausgangs-Systemvariablen 240
Tabelle 241: Empfangsvariablen 240
Tabelle 242: A-Eingänge Funktionsbaustein TCP_ReceiveVar 243
Tabelle 243: A-Ausgänge Funktionsbaustein TCP_ReceiveVar 244
Tabelle 244: F-Eingänge Funktionsbaustein TCP_ReceiveVar 244
Tabelle 245: F-Ausgänge Funktionsbaustein TCP_ReceiveVar 244

Tabelle 246: Eingangs-Systemvariablen 245
Tabelle 247: Ausgangs-Systemvariablen 245
Tabelle 248: Empfangsvariablen 245
Tabelle 249: Anzeigefeld S&R Protokoll 247
Tabelle 250: Anzeigefeld Modbus Slaves 247
Tabelle 251: Fehlercodes der TCP-Verbindung 248
Tabelle 252: Zusätzliche Fehlercodes 249
Tabelle 253: Verbindungszustand 249
Tabelle 254: Partner Verbindungszustand 249
Tabelle 255: Systemanforderung und Ausstattung S&R TCP 250
Tabelle 256: Eigenschaften des SNTP-Client 251
Tabelle 257: Eigenschaften SNTP-Server-Info 252
Tabelle 258: Eigenschaften SNTP-Server 253
Tabelle 259: Systemanforderung und Ausstattung des X-OPC-Server 254
Tabelle 260: Eigenschaften des X-OPC-Server 255
Tabelle 261: Erforderliche Aktionen bei Änderungen 257
Tabelle 262: Standardwerte der Prioritäten 269
Tabelle 263: Zustand und der Zeitstempel der Data Access Views 271
Tabelle 264: Parameter für boolesche Ereignisse 273
Tabelle 265: Parameter für skalare Ereignisse 275
Tabelle 266: Eigenschaften 279
Tabelle 267: Eigenschaften 280
Tabelle 268: Edit 280
Tabelle 269: Systemanforderung und Ausstattung ComUserTask 281
Tabelle 270: Eigenschaften ComUserTask 281
Tabelle 271: Abkürzungen 282
Tabelle 272: Schedule-Intervall [ms] 283
Tabelle 273: Systemvariablen der ComUserTask 285
Tabelle 274: Eingangssignale des ComUserTask 286
Tabelle 275: Ausgangssignale des ComUserTask 286
Tabelle 276: Parameter 288
Tabelle 277: Parameter 290
Tabelle 278: Rückgabewert 290
Tabelle 279: Parameter 291
Tabelle 280: Rückgabewert 291
Tabelle 281: Parameter 292
Tabelle 282: Rückgabewert 293
Tabelle 283: Parameter 294
Tabelle 284: Parameter 295
Tabelle 285: Rückgabewert 295
Tabelle 286: Parameter 296
Tabelle 287: Parameter 297
Tabelle 288: Rückgabewert 297
Tabelle 289: Rückgabewert 298
Tabelle 290: Parameter 299
Tabelle 291: Parameter 300
Tabelle 292: Rückgabewert 300
Tabelle 293: Parameter 301
Tabelle 294: Parameter 302
Tabelle 295: Parameter 303
Tabelle 296: Rückgabewert 303
Tabelle 297: Parameter 304
Tabelle 298: Parameter 305
Tabelle 299: Rückgabewert 305
Tabelle 300: Parameter 306
Tabelle 301: Rückgabewert 306
Tabelle 302: Parameter 307
Tabelle 303: Parameter 308
Tabelle 304: Rückgabewert 308
Tabelle 305: Parameter 309
Tabelle 306: Parameter 310
Tabelle 307: Rückgabewert 310

Tabelle 308: Parameter 312
Tabelle 309: Befehle in Cygwin (Bash Shell) 315
Tabelle 310: Ausgangsvariablen (CPU->COM) 325
Tabelle 311: Eingangsvariablen (COM->CPU) 325

Index

Einsatzbedingungen
EMV 15
ESD-Schutz 16
klimatisch 14
mechanisch 15
Spannungsversorgung 16
Teilenummer 21

HI 801 100 D
© 2009 HIMA Paul Hildebrandt GmbH + Co KG
HIMax und SILworX sind registrierte Warenzeichen von:
HIMA Paul Hildebrandt GmbH + Co KG

Albert-Bassermann-Str. 28
68782 Brühl, Deutschland
Tel. +49 6202 709-0
Fax +49 6202 709-107
HIMax-info@hima.com
www.hima.com

HIMA

SAFETY
NONSTOP